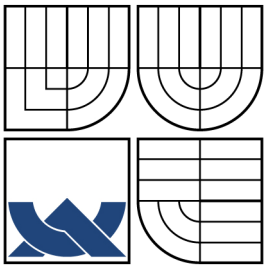
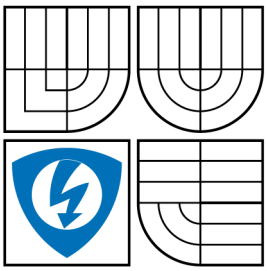


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

# SOFTWAROVÁ PODPORA VÝUKY KRYPTOSYSTÉMŮ ZALOŽENÝCH NA PROBLÉMU DISKRÉTNÍHO LOGARITMU

SOFTWARE SUPPORT FOR CRYPTOGRAPHY SYSTEM TRAINING BASED ON DISCRETE LOGARITHM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. JIŘÍ KŘÍŽ

VEDOUCÍ PRÁCE
SUPERVISOR
Doc. Ing. KAREL BURDA CSc.

BRNO 2008

ORIGINÁL ZADÁNÍ

## ANOTACE

Potřeby mezilidské komunikace v současné době dospěly do stavu, kdy většina přenášených zpráv je důvěrné povahy a jejich přenos po sdílených nezabezpečených linkách v otevřené podobě není možný. Z toho důvodu vzniklo velké množství metod pro šifrování zpráv a přenos v zabezpečené podobě. Vytvořily se dva hlavní vývojové proudy, symetrická a asymetrická kryptografie.

Druhá zmíněná skupina je založena na využití dvou informací – klíčů, kdy jeden je veřejně znám a druhý je tajný. Použitím veřejného klíče lze snadno určit kryptogram zprávy, k jeho dešifrování je však třeba znát tajný klíč. Tyto metody jsou založeny na matematických problémech, pro které současná matematika nezná časově efektivní algoritmus výpočtu.

Práce se zaměřuje na kryptosystémy, založené na problému diskrétního logaritmu, kdy šifrování zpráv lze provést z veřejně známých parametrů – veřejného klíče velmi rychle, ale dešifrování bez znalosti tajné informace – tajného klíče, je časově extrémně náročné.

Je zde rozebrán samotný matematický problém diskrétního logaritmu, jeho vlastnosti a metody, které se jej snaží řešit. Popsána je také komunikace s využitím kryptosystémů na diskrétním logaritmu založených, jako ElGamalův kryptosystém, Diffie-Hellmanův protokol nebo DSA.

Druhá část práce se pak zaměřuje na webovou aplikaci vytvořenou pro podporu výuky problému diskrétního logaritmu a kryptosystémů na něm založených. Popisuje jak funkční a grafické rozhraní, tak práci s ním a možnosti, které uživateli nabízí. Obsahuje také úkoly, které by měly pomoci uživateli v pochopení dané problematiky a k jejímu procvičení.

## KLÍČOVÁ SLOVA

Diskrétní logaritmus, cyklická grupa, asymetrický kryptosystém, NP problém, Diffie-Hellmanův protokol, DSA, ElGamal, hash

**Bibliografická citace mé práce:**

KŘÍŽ, J. Softwarová podpora výuky kryptosystémů založených na problému diskrétního logaritmu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 74 s. Vedoucí diplomové práce doc. Ing. Karel Burda, CSc.

## ABSTRACT

Current needs of human communication came to status, when most of transferred messages are considered as private and transition over non-secured communication lines in open form is not possible. That originated a lot of different methods for securing of messages and transfers in ciphered form. Two mainstreams were established, symmetric cryptography and asymmetric cryptography.

Second of mentioned groups is based on usage of two information – keys, when one of then is broadly known and is public and second, well protected and private. Using a public key it is possible to establish a cryptogram of message, but for deciphering it is necessary to know private key. Asymmetric methods are based on mathematical problems, for which there is not an effective computing algorithm.

This thesis are focused for asymmetric cryptosystems based on discrete logarithm problem, where ciphering of message using public key is very easy and quick, but deciphering without knowledge of private key is extremely time consuming process.

Work describes a mathematical base of discrete logarithm problem, its' properties and methods developed for solving of this problem. Descriptions of particular cryptosystems are given, i.e. ElGamal cryptosystem, Diffie-Hellman protocol and DSA.

Second part of thesis is focused for web application developed as study support of discrete logarithm problem and of cryptosystems using this problem. It describes functional and graphical interface, work with it and options given to user working with application. Mentions also lessons for user which should help with understanding of described problems and practicing.



## KEY WORDS

Discrete Logarithm, Cyclic Group, Asymmetric Cryptosystem, NP problem, Diffie-Hellman Protocol, DSA, ElGamal

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma "Softwarová podpora výuky kryptosystémů založených na problému diskrétního logaritmu" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……..

(podpis autora)

---

# PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Doc. Ing. Karlu Burdovi CSc., za užitečnou metodickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne …… ……..

(podpis autora)

# Obsah

1 ÚVOD ..... 1
1.1 Vývoj kryptografie ..... 2
2 KRYPTOLOGIE A JEJÍ POJMY ..... 3
2.1 Symetrická kryptografie ..... 3
2.2 Asymetrická kryptografie ..... 4
2.2.1 Komunikace v asymetrické kryptografii ..... 5
3 MATEMATICKÉ PROBLÉMY A JEJICH VYUŽITÍ ..... 8
3.1 P, NP a NP-úplné problémy ..... 8
3.2 Problém faktorizace celých čísel ..... 9
3.2.1 RSA ..... 9
3.3 Problém výpočtu kvadratických reziduí ..... 10
3.3.1 Rabinův kryptosystém ..... 10
3.4 Problém diskrétního logaritmu ..... 11
3.4.1 Cyklická grupa G ..... 12
3.4.2 Vlastnosti diskrétního logaritmu ..... 13
3.4.3 Metoda „baby-step giant-step“ ..... 14
3.4.4 Algoritmus indexového výpočtu ..... 15
3.4.5 Pollardův rho algoritmus pro logaritmy ..... 15
4 KRYPTOSYSTÉMY ZALOŽENÉ NA PROBLÉMU DISKRÉTNÍHO LOGARITMU 17
4.1 Diffie-Hellmanův protokol ..... 17
4.1.1 Bezpečnost Diffie-Hellmanova protokolu ..... 18
4.2 ElGamalův algoritmus ..... 19
4.2.1 ElGamalův kryptosystém pro šifrování zpráv ..... 20
4.2.2 ElGamalův systém pro podepisování zpráv ..... 22
4.2.3 Bezpečnost ElGamalova kryptosystému ..... 24
4.3 Hashovací funkce založená na diskrétním logaritmu ..... 25
4.3.1 Hash zprávy ..... 25
4.3.2 Shamirova hashovací funkce ..... 25
4.4 DSA – Digital Signature Algorithm ..... 27
4.5 Obecná bezpečnost kryptosystémů DL ..... 28
4.5.1 Postranní kanály v kryptografii ..... 28
5 APLIKACE PRO PODPORU VÝUKY ..... 30

5.1 Aplikační platforma ..... 30
5.2 Programové prostředí Java ..... 30
5.2.1 Modulo velkých čísel ..... 31
5.3 Struktura aplikace ..... 32
5.4 Aplikační server ..... 33
7 ROZHRANÍ WEBOVÉ APLIKACE ..... 35
7.1 Základní nabídka ..... 35
7.2 Generátor prvočísel ..... 35
7.3 Průvodce ..... 36
7.4 Test znalostí ..... 37
7.5 Modul „Diskrétní logaritmus“ ..... 37
7.6 Modul „ElGamalův kryptosystém – šifrování“ ..... 38
7.7 Modul „ElGamalův kryptosystém – podepisování“ ..... 40
7.8 Modul „Diffie-Hellmanův protokol“ ..... 41
7.9 Modul „DSA a podepisování zpráv“ ..... 42
8 ZÁVĚR ..... 43
POUŽITÁ LITERATURA ..... 45

# Seznam obrázků

Obr.1: Zjednodušený přenos dat v asymetrické kryptografii ..... 5
Obr.2: Přenos dat v asymetrické kryptografii ..... 6
Obr. 3: Průběh hodnot diskrétního mocnění $b^k$ mod n v multipl. grupě řádu n=107 ..... 14
Obr. 4: Ustavení klíče pomocí Diffie-Helmanova protokolu ..... 18
Obr. 5: Přenos šifrované zprávy protokolem ElGamal ..... 22
Obr. 7: Propojení dílčích částí aplikace ..... 33
Obr. 8: Základní menu aplikace ..... 35
Obr. 9: Blok generátoru prvočísel ..... 36
Obr. 10: Grafické rozhraní průvodce ..... 36
Obr. 11: Testové grafické rozhraní ..... 37
Obr. 12: Výpočet klíče ElGamalova kryptosystému ..... 39
Obr. 13: Rozhraní ElGamalova systému pro šifrování zpráv ..... 39
Obr. 14: Rozhraní Shamirova hashovacího algoritmu ..... 40
Obr. 15: Grafické rozhraní modulu Diffie-Hellmanova protokolu ..... 41

# 1 ÚVOD

Lidská komunikace od dávných dob spočívá ve vzájemné výměně informací. Od té doby existuje také potřeba omezení přístupu k vyměňovaným informacím pouze pro vybraný okruh lidí. Za tímto účelem tak vzniklo velké množství nejrůznějších technik, které přenášené informace upravují do formy, která je čitelná pouze určenému okruhu lidí. Tento okruh je vyčleněn speciální znalostí, kterou musí disponovat. Typicky je tato znalost označována jako šifrovací/dešifrovací klíč. Současně existuje skupina lidí, kteří ačkoli nepatří mezi osoby s příslušným klíčem, věnuje svoji snahu získání skryté informace jiným způsobem. Těmto problémům se věnuje samostatná vědní disciplína, kryptologie. Je to věda, jejíž obory kryptografie a kryptoanalýza se zabývají způsoby skrývání informace a způsoby, jak tuto informaci získat zpět do čitelné formy bez znalosti informací použitých pro skrytí informace.

V průběhu času vzniklo velké množství technik a přístupů, které slouží k zabezpečení, šifrování, informace. Obecně je lze rozdělit do dvou velkých skupin na základě způsobu, jakým pracují se svým klíčem. Jde o metody symetrické a asymetrické. Asymetrickým technikám, resp. jejich podskupině založené na matematickém problému diskrétního logaritmu, bude věnována většina této práce.

Cílem projektu je představit základní matematické problémy, kterých asymetrická kryptografie využívá v různých metodách šifrování. Hlavní pozornost bude věnována problému diskrétního logaritmu, jehož problematické řešitelnosti se využívá hned v několika různých kryptosystémech. Práce si klade za cíl umožnit pochopení problému diskrétního logaritmu a techniky, kterými kryptosystémy k tomuto problému přistupují. Projekt na příkladech demonstruje využití diskrétních logaritmů pro šifrování dat, podepisování zpráv a nebo k ustavení klíče využitelného některou ze symetrických kryptografických metod.

Praktická část projektu se zaměřuje na návrh webové aplikace pro podporu výuky kryptosystémů založených na diskrétním logaritmu. Jejím cílem je představit samotný diskrétní algoritmus jako matematický úkol, poskytnout nástroj pro jeho řešení a naznačit vlastnosti cyklických grup, ve kterých spočívá jeho obtížnost. Dále pak představuje kryptosystémy z diskrétního logaritmu vycházející, poskytuje názornou ukázku jejich funkce a vlastností. Doprovodný text spustitelný z aplikace pak obsahuje jednoduché úkoly, které mají uživateli aplikace zajistit lepší zapamatování problematiky a její procvičení na praktických příkladech.

Práce představuje jak dílčí rozhraní aplikace a práci s ním, tak částečný pohled do implementace problému v jazyce Java pro potřeby případných modifikací v budoucnu.

## 1.1 Vývoj kryptografie

Počátky kryptografie sahají poměrně hluboko do lidské minulosti. Jako první se kryptografická metoda se začalo používat úprav podob písma, nešlo o příliš sofistikovanou metodu, ale jelikož schopnost psát a číst nebyla vůbec běžnou, stačila taková úprava pro znečitelnění zprávy. V pozdějších dobách se začalo používat prohazování znaků podle různých schémat. Existovala například technika, kdy se text psal na tenký pruh papíru natočený na válci o určitém průměru, kdy pro přečtení bylo potřeba znát správný průměr válce. V dobách „Viktoriánské“ Anglie se oblíbenou kryptografickou metodou stalo nahrazení jednotlivých znaků abecedy jinými znaky téže abecedy nebo jiné abecedy o odpovídajícím počtu znaků. Do doby objevení statistického útoku, kdy se spočte průměrný počet znaků v libovolném otevřeném textu daného jazyka a přiřadí se odpovídajícímu výskytu znaku v šifrovaném textu, šlo o velmi zdařilou šifru. Šifrovací techniky se různě kombinovaly, například i se steganografickými metodami (zjednodušeně jde o uschování zprávy tak, aby nebylo poznat, že se nějaká zpráva přenáší).

Postupným vývojem techniky vznikly šifrovací mechanismy, které dokázaly odesílanou zprávu šifrovat a s příslušným klíčem luštit v reálném čase. Šlo například o slavný přístroj Enigma, který vymyslelo Německo v průběhu 2. světové války. Tato válka znamenala v oblasti kryptografie velký pokrok, k ještě masivnějšímu rozvoji však došlo v době tzv. „studené války.“ V této době již byla známa spousta technik, včetně „dokonalé šifry“ o které bude ještě zmínka. Hlavní problém té doby představovala bezpečná distribuce šifrovacího klíče. Tato slabina symetrických kryptosystémů byla natolik významnou, že snaha o vytvoření lepších systémů v žádném případě neutichala. Řešení přišlo se vznikem asymetrických kryptosystémů, které sice stejně jako všechny předchozí zmíněné techniky nabídly pouze reálné utajení (ze všech klíčů, které lze v dané abecedě vytvořit, pouze použití jediného konkrétního klíče k dešifrování dává smysluplnou zprávu), ale zajistily, že dešifrovací klíč se nemusí přenášet po nezabezpečených trasách.

# 2 KRYPTOLOGIE A JEJÍ POJMY

Kryptologie je vědní disciplína dělící se na dva hlavní obory, kryptografii a kryptoanalýzu. Kryptografie je věda zabývající se převodem informace do podoby čitelné pouze se znalostí tajného klíče, kryptoanalýza je proti tomu věda o převodu šifrované/tajné informace do srozumitelné/otevřené podoby.

Současná kryptografie je většinou založená na použití různých matematických aparátů a problémů, které lidstvo při současných znalostech a daném strojovém výpočetním výkonu nedokáže při splnění určitých podmínek řešit v reálném čase. Dostatečnou podmínkou je ve většině případů dostatečná délka šifrovacího klíče a dodržení podmínek pro jeho tvorbu. Vzhledem k relativně malému počtu šifrovacích algoritmů se nepřistupuje k utajení použitého algoritmu, ale pouze klíče. Jeho bezpečnost je pak naprosto zásadní. Kryptografii je možné rozdělit na dvě hlavní skupiny, symetrickou a asymetrickou. Toto rozdělení je dáno způsobem, kterým metody hospodaří s šifrovacím klíčem. Velmi významný rozdíl mezi nimi však také tvoří výpočetní náročnost a s ní související rychlost, jako jsou schopny šifrovat data. Zatímco symetrické algoritmy dokáží data šifrovat s velkou rychlostí, asymetrické šifrování je velmi pomalé a pro velké objemy dat se tedy nehodí. Obě metody se tedy často kombinují tak, že asymetrickou kryptografií se bezpečně přenese šifrovací klíč pro symetrickou kryptografii, kterou se poté přenáší potřebná data.

Kryptoanalýza je velmi rozsáhlou vědou. Její metody lze částečně také rozdělit, ale vzhledem k tomu, že většinou se každá analýza (útok) přizpůsobuje přesně konkrétním podmínkám a metod se kombinují, není toto rozdělení příliš účelné. V dřívějších dobách se kryptoanalýza prováděla především zkoumáním matematických vlastností šifrovacího algoritmu. Dnešní šifrovací algoritmy však matematicky prolomitelné nejsou (v reálném čase, kdy má toto prolomení ještě nějaký význam) a proto se přechází ke zkoumání konkrétní implementace na fyzickém zařízení. Tento přístup přináší v posledních letech velmi dobré výsledky a daří se s jeho pomocí rozluštit i algoritmy, které se z matematických hledisek zdají neprolomitelné. Podrobnější informace lze získat např. v [1].

## 2.1 Symetrická kryptografie

První skupinou kryptografických metod jsou symetrické algoritmy. Ty využívají pro svoji funkci šifrování a dešifrování vzájemně odvoditelné klíče. Vstupem algoritmu jsou data

zapsaná pomocí stanovené abecedy (v současné době je to nejčastěji abeceda binární) a zvolený klíč. Šifrovací algoritmus na straně odesilatele vypočte ze zmíněných vstupů kód, který je možné přenášet po veřejných datových spojích, kde je přístupný „všem.“ Příjemce po obdržení tohoto kódu využije odpovídající dešifrovací algoritmus a stejný nebo vhodně modifikovaný klíč jako odesilatel k opětovnému získání původního utajeného textu (plaintext).

Při šifrování se využívá algoritmů, u nichž při znalosti vstupního i zakódovaného textu je velmi obtížné získat použitý šifrovací klíč. Při jeho znalosti je ale kódování i dekódování záležitost víceméně velmi snadná a rychlá. Obtížnost získání klíče jako taková je převážně dána jeho délkou, přičemž platí, že čím je klíč delší, tím je jeho získání obtížnější. Zakódovaná zpráva musí „odolat“ i útoku „brute force,“ který předpokládá vyzkoušení všech kombinací klíčů k prolomení šifry. Pojem odolat zde znamená, že útok by trval natolik dlouho, že jej není možné provést v takovém čase, kdy by výsledky byly stále užitečné.

Ze dříve zmíněného plyne, že ideální šifrovací klíč bude stejně dlouhý jako zpráva určená k utajení, bude náhodný a současně bude použit jen jednou, což jsou základní podmínky tzv. „Dokonalé šifry,“ kterou nelze žádným způsobem prolomit. Takový klíč je však velmi obtížné získat a především distribuovat mezi uživatele. Proto se takových klíčů a šifer v praxi nepoužívá. V současnosti používané klíče mají délky 128 bitů nebo více, čemuž odpovídá $2^{128}$ možných kombinací klíče. Pro představu, vyzkoušení všech těchto klíčů při útoku „Brute force“ by při zkoušení $1.10^9$ klíčů za vteřinu trvalo přibližně $1.10^{22}$ let.

Symetrické metody kódování obsahují jednu, již zmíněnou, nevýhodu. Tou je problém, jak technicky zajistit distribuci klíče mezi odesilatele a příjemce dat takovým způsobem, aby se tento klíč nemohl dostat do nepravých rukou. [1].

## 2.2 Asymetrická kryptografie

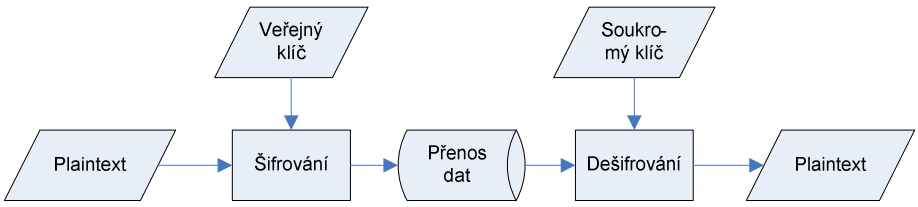
Problém distribuce klíče odstraňují metody tzv. asymetrické kryptografie. Asymetrické proto, že pro kódování a dekódování využívá odlišných klíčů. Uživatel pro svoji práci vytvoří dva klíče, které mezi sebou mají vztah definovaný konkrétním šifrovacím algoritmem. Přitom na rozdíl od symetrické kryptografie platí, že klíče nejsou vzájemně odvoditelné (za znalosti veřejného není jednoduše možné určit soukromý). Jeden takto vytvořený klíč si uživatel bezpečně uschová, označujeme jej jako „soukromý klíč,“ a druhý libovolným způsobem zveřejní. Tento klíč se označuje jako „veřejný klíč.“

U asymetrických šifer se délka klíče výrazně liší od symetrických metod, protože se zde pracuje většinou pouze s užším oborem čísel, jako jsou například prvočísla. Pokud by délka klíče byla stejná, stačilo by při pokusech o prolomení šifry zkoumat pouze kombinace v tomto úzkém oboru hodnot a doba potřebná k prolomení šifry by neúnosně klesla. Proto se v současné době běžně pracuje s klíči délek 1024 nebo 2048 bitů.

Asymetrické kryptografické metody dále využívají při svojí funkci takzvaných jednocestných funkcí, hashů, což jsou funkce jež lze relativně snadno vyčíslit, ale je prakticky nemožné z jejich výsledku určit původní vstup bez znalosti původních vstupních informací. Hashovací metody budou v krátkosti představeny později a bude rozebrán i jejich praktický význam.

### 2.2.1 Komunikace v asymetrické kryptografii

Nyní bude účelné podrobněji vysvětlit, jak vlastně asymetricky zabezpečená komunikace probíhá. Výchozí myšlenka již byla naznačena. Uživatel vygeneruje dva klíče, jeden si uschová a druhý vhodným způsobem zveřejní. V nejjednodušším případě, pokud někdo chce tomuto uživateli (příjemci) poslat zprávu, vezme jeho veřejný klíč, který někde získal, a zašifruje s jeho pomocí zasílaná data. K dešifrování těchto dat je třeba použít druhého klíče z původní dvojice klíčů. Tento klíč má uschován pouze příjemce a jelikož se klíč nikde nepřenášel po nezabezpečených trasách, lze jej považovat za bezpečný. Data je tedy schopen dešifrovat pouze oprávněný příjemce. Ostatní uživatelé sice mohou zkoušet rozšifrovat data, ale jelikož neznají správný klíč, pravděpodobnost že se jim to podaří je velice malá. Takovouto komunikaci znázorňuje obr.1.




*Obr.1: Zjednodušený přenos dat v asymetrické kryptografii*

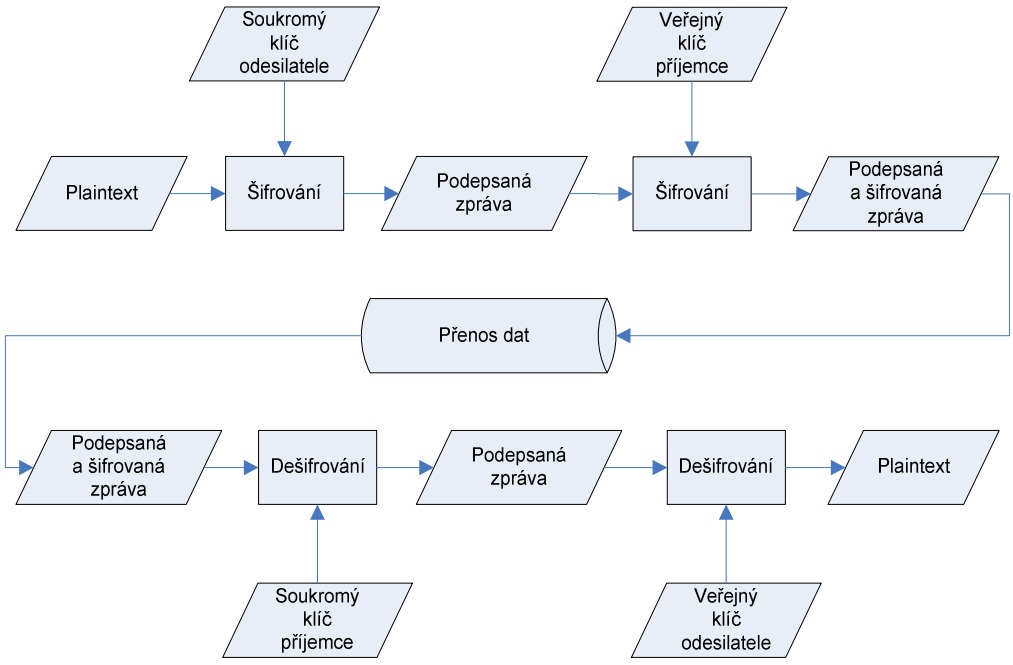
Tato situace je velice zjednodušená. Příjemce dat v tomto případě nemá žádnou záruku, že data odeslal opravdu ten, od koho je příjemce předpokládal, protože jeho veřejný klíč může znát kdokoli. Pokud by útočník při přenosu data zachytil, nebyl by sice schopen je dešifrovat, ale mohl by místo nich veřejným klíčem zašifrovat data vlastní a ty odeslat nic netušícímu

příjemci (část útoku „Man in the middle"). Pro zajištění autorizace odeslaných dat se používá technika podepisování.

Podepisování je technika, kdy odesílatel zprávy tuto zprávu zašifruje svým vlastním soukromým klíčem. Význam podpisu spočívá v tom, že kdokoli je schopen data dešifrovat s použitím příslušného veřejného klíče, ale nikdo není schopen místo zprávy podvrhnout zprávu jinou, protože nezná klíč, kterým byla zpráva původně zašifrována. Tím je zajištěna autorizace zprávy. Pouze předpokládaný odesilatel ji byl schopen zašifrovat správným klíčem.

Pro účely podepsání zprávy většinou nešifruje celá zpráva, ale pouze její jednosměrný otisk (hash), který vznikne zvoleným algoritmem z původní zprávy. Tento hash je zpravidla mnohonásobně menší než původní zpráva a jeho zašifrování relativně pomalými algoritmy asymetrické kryptografie je tak rychlejší. Hash navíc zajistí autentičnost zprávy. Podrobné informace o hashovacích funkcích jsou nad rámec tohoto textu, případné zájemce lze odkázat na [2] nebo kapitolu 4.3.1.

Komunikace pak s použitím techniky podepisování vypadá tak, že odesílatel zprávu nejprve zašifruje svým soukromým klíčem (podepíše ji) a následně ji zašifruje ještě veřejným klíčem příjemce. Příjemce po obdržení zprávy tuto zprávu dešifruje nejprve svým soukromým klíčem a následně ověří pravost zprávy dešifrováním pomocí veřejného klíče odesílatele. Takováto komunikace je naznačena na obrázku 2.




*Obr.2: Přenos dat v asymetrické kryptografii*

Popsaný postup je již zcela bezpečný, ale pouze v případě, že odesílatel i příjemce znají vzájemně své veřejné klíče. Pokud by tomu tak nebylo, útok „Man in the middle" by byl stále možný tak, že by útočník podvrhl své vlastní klíče uživatelům a mohl by tedy podvrženou zprávu podepsat vlastním klíčem.

Z tohoto důvodu byla ustavena infrastruktura veřejných klíčů. Jde o systém, kdy uživatel s vytvořeným veřejným klíčem přijde k obecně uznávané a důvěryhodné certifikační autoritě, prokáže svoji identitu a skutečnost, že k předloženému veřejnému klíči zná klíč soukromý. Certifikační autorita pak svým soukromým klíčem podepíše hash údajů o uživateli svým soukromým klíčem. Tento podpis spolu s údaji o uživateli a jeho veřejným klíčem pak tvoří „certifikát.“ Veřejný klíč certifikační autority je dobře známý a každý tedy může pravost podpisu snadno ověřit. Navíc zmíněná infrastruktura obsahuje i registr s klíči, které jsou formálně správné ale jejich vlastníci je již z důvodu ztráty bezpečnosti nepoužívají.

Spolu s podepsanou zprávou se tedy přenáší také certifikát, který jednoznačně identifikuje osobu odesilatele. Více o podepisování zpráv v kapitolách 4.2.2 a 4.4.

# 3 MATEMATICKÉ PROBLÉMY A JEJICH VYUŽITÍ

Matematických problémů, které současná matematika nedokáže efektivně řešit v současnosti existuje velká řada, některé jsou spíše jen teoretického řádu, jiné nacházejí velké uplatnění v různých lidských činnostech. Pro jejich popis je vhodné je určitým způsobem rozdělit. Nejběžnějším rozdělením je rozdělení podle složitosti do tříd P, NP a NP-úplné.

## 3.1 P, NP a NP-úplné problémy

Náležitost jednotlivých problémů do těchto skupin je poměrně problematická a řadí se k velkým problémům v oblasti výpočtů složitosti. Teorie složitosti do těchto skupin přiřazuje problémy podle toho, jak náročný výpočet je nutný, aby byl problém vyřešen. Skupina P zahrnuje problémy, jejichž složitost lze zapsat jako polynomiální funkci. Čas nutný pro výpočet takových problémů je pak deterministicky polynomiální, někdy se proto takové úlohy označují jako lehké [3]. Do skupiny NP spadají problémy jejichž funkce jsou například exponenciální (mocninné) nebo dokonce faktoriální. Čas nutný pro výpočet je nedeterministicky polynomiální, tedy se změnou počítaných parametrů velice rychle narůstá. Poslední skupinou jsou problémy NP-úplné. Jde o podmnožinu NP problémů, které jsou považovány za nejsložitější. Patří sem například problém faktorizace velkých čísel, problém diskrétního logaritmu ale také například problém výběru nejlepšího tahu v šachu. Jde o problémy, kde lze snadno ověřit správnost známého řešení, ale je velmi obtížné takové řešení spočítat.

Z matematického hlediska jsou problémy NP (NP-úplné) ekvivalentní s P problémy. Je tedy teoreticky možné NP problémy převést na P problémy a řešit je v „krátkém“ čase. Jelikož jsou tyto problémy ekvivalentní, nalezení řešení jediného z těchto problémů umožní řešit i všechny ostatní NP problémy. Pokud se tedy podaří takové řešení nalézt, bude to pravděpodobně konec asymetrické kryptografie tak jak existuje v současnosti. Finanční hodnota takového řešení je proto obrovská a pokusy o jeho nalezení stále probíhají, stejně jako existuje snaha o prokázání, že takové řešení stále neexistuje.

## 3.2 Problém faktorizace celých čísel

Faktorizace je proces, při kterém je snaha rozložit určité zadané číslo na součin dvou prvočísel. Jedná se o úlohu patřící mezi NP-úplné problémy a při současné úrovni matematiky lze řešit pouze hrubou silou, tj. postupným výčtem všech možností. Postup je tedy takový, že se postupně dělí stále většími a většími čísly do doby, než se najde výsledek. Vynásobit mezi sebou dvě i značně velká čísla není příliš obtížné, naopak provést rozklad na součin v reálném čase je pro velká čísla mimo možnosti současné výpočetní techniky. Právě z této asymetrie v rozdílu těchto obtížností vychází podstata použití v kryptografii [4].

Stále ovšem probíhá snaha o faktorizaci co největších čísel, která je podporována finančními granty od společností jako RSA Laboratories, které tímto způsobem dokazují odolnost svých produktů, hlavně tedy šifrovacího a podepisovacího systému RSA. V roce 2005 tak bylo například úspěšně faktorizováno číslo o 663-ti bitech délky a v současnosti díky internetovým projektům, do kterých jsou zapojeny stovky počítačů, se pracuje na číslech ještě vyšších.

### 3.2.1 RSA

RSA je jedním z nejznámějších šifrovacích algoritmů využívajících veřejného klíče. Vznikl v roce 1977 spoluprací pánů Rivesta, Shamira a Adlemana a v roce 1983 byl patentován v USA [5]. Jeho bezpečnost je založena na problému faktorizace a RSA problému, který je definován jako úloha získání *e*-tého kořene modulu množiny *n*, obnovení hodnoty *m* jako $m^e = c \bmod n$, kde (*n*, *e*) jsou veřejným klíčem RSA a *c* je šifrovaný text.

Tvorba veřejného klíče pobíhá v několika krocích:

1) Zvolí se dvě prvočísla *p* a *q*. Prvočísla by neměla být příliš blízkých hodnot. Vypočte se jejich součin N, který tvoří část veřejného klíče.
2) Zvolí se další číslo *e*, pro něž platí $e \leq (p-1).(q-1)$ a je s tímto součinem nesoudělné. Jde též o prvočíslo a tvoří druhou část veřejného klíče.

Šifrování probíhá dle následujícího vzorce:

$C = M^e \pmod N$, kde M je zpráva určená k šifrování

Dešifrování:

Nejdříve je třeba určit *d* podle: $ed = 1 \pmod{(p-1)(q-1)}$

$C^d \pmod N = M$

V současné době se za bezpečnou délku klíče považuje délka 2048 bitů. Vzhledem k nízké rychlosti šifrování, kterou se vyznačují všechny asymetrické kryptosystémy, se používá především pro přenos klíčů symetrické kryptografie a podepisování zpráv. Bližší popis RSA a všech jeho vlastností je již nad rámec tohoto textu, další informace lze získat např. v [6].

## 3.3 Problém výpočtu kvadratických reziduí

Jedním z dalších matematických problémů patřících do skupiny NP-úplných problémů je problém výpočtu kvadratických reziduí. V matematice se číslo $q$ nazývá kvadratické reziduum modulo $n$, pokud existuje celé číslo $x$ $(0 < x < n)$, pro které platí

$$x^2 \equiv q \pmod{n}.$$

Pokud takové číslo x neexistuje, pak se q nazývá kvadratické nonresiduum (mod $n$). Bylo dokázáno, že jednotlivá rezidua se v zadaném intervalu objevují náhodně. Vlastností reziduí pak využívá Rabinův kryptosystém.

### 3.3.1 Rabinův kryptosystém

Jde o asymetrický kryptosystém založený na výpočtu kvadratických reziduí. Jeho bezpečnost je, stejně jako bezpečnost RSA, podložena výpočtem faktorizace celých čísel. Oproti RSA však bylo dokázáno, že jeho bezpečnost je stejná, jako obtížnost faktorizace, což u RSA kryptosystému dokázáno nebylo. Jeho bezpečnost je však také zpochybňována, zejména díky jeho náchylnosti na „chosen-ciphertext attack" [7].

Postup při tvorbě klíčů je (zjednodušeně) následující:

1) zvolí se dvě dostatečně velká prvočísla $p$ a $q$
2) vypočte se jejich součin $n = p.q$, který tvoří veřejný klíč

Šifrování pak probíhá podle vztahu

$$c = m^2 \bmod n,$$

kde $m$ je příslušná část zprávy. Dešifrování probíhá ve více krocích, jejich popis přesahuje rámec tohoto textu. Podrobný popis je lze nalézt například v [7].

Důležitou a velmi nevýhodnou vlastností tohoto šifrovacího algoritmu je skutečnost, že stejný šifrovaný obraz $c$ může vzniknout z množiny až čtyř různých vstupních dat $m$. Po dešifrování je tedy výsledkem tato množina a je nutné rozhodnout, který prvek byl součástí

původních dat. Toto je poměrně snadné pro textová data, kde jen jediná zpráva bude dávat smysl, ale velmi obtížné například pro grafický obraz nebo čísla.

## 3.4 Problém diskrétního logaritmu

V kryptografické praxi velmi často užívaným matematickým problémem je problém diskrétního logaritmu. Jde opět o NP-úplný problém, tedy se předpokládá, že jeho řešení nelze nalézt v polynomiálním čase.

K vysvětlení podstaty tohoto problému je třeba pochopit některé pojmy. Prvním je definice logaritmu jako takového: $y=log_a\ x$, ze kterého výpočet diskrétního logaritmu vychází. Tento logaritmus je opakem mocninné funkce $x=a^y$ v oboru reálných i imaginárních čísel.

Dalším nutným pojmem je konečná cyklická grupa (skupina) $G$ $(\mathbf{Z}_n)^x$. Vzhledem k zásadní důležitosti tohoto pojmu, je tématu věnována samostatná kapitola 3.4.1.

Diskrétní logaritmus je tedy definován tak, že je-li $n$ prvočíslo a $b$, $g \in (\mathbf{Z}_n)^x$, pak diskrétním logaritmem čísla $g$ je $k$ a lze psát

$$log_b(g)=k,$$

kde $k$ je celé číslo splňující podmínku $b^k=g$. Nalezení takového celého čísla $k$ pro daná $b, g \in (\mathbf{Z}_n)^x$, (když $b \neq 1$) se nazývá Problém diskrétního logaritmu [8][9]. Ve skutečnosti diskrétní logaritmus $k$ není jedinečný, protože lze nalézt pouze jeho hodnotu jako zbytek po celočíselném dělení hodnotou (řádem) $n$ cyklické skupiny, takže platí

$$log_b(g)=k\ (mod\ n).$$

Pro lepší pochopení diskrétního logaritmu poslouží následující příklad.
Nejprve se zvolí skupina $(\mathbf{Z}_n)^x$ tvořená řadou čísel od $1$ do $n-1$, kde $n$ je celočíselný dělitel (modulo $n$), jde zpravidla o prvočíslo.

Na začátku je vhodné ukázat diskrétní mocnění v této cyklické skupině. Zvolí se například skupina $(\mathbf{Z}_{13})^x$. Bude-li se hledat $k$-tá mocnina čísla $b$ v této skupině, pak se toto číslo nejprve umocní na $k$ a poté se vypočte zbytek po celočíselném dělení číslem $n$. Například tedy $15^5 = 6 \pmod{13}$ ve skupině $(\mathbf{Z}_{13})^x$. Tento výpočet není nikterak náročný a lze ho provést velmi rychle.

Výpočet diskrétního logaritmu je, jak z úvodu vyplývá, opačnou operací k diskrétnímu mocnění. Je-li dána situace, kdy $15^k \equiv 6 \pmod{13}$, pak pro výpočet $k$ neexistuje žádný efektivní algoritmus který by vedl k řešení. Z předešlého výpočtu je vidět, že řešením je $k$=5,

nejde ale o jediné řešení. Z vlastností cyklické multiplikativní skupiny plyne, že platí $15^{12} = 1$ (mod 13), lze tedy psát $15^{5+12x} = 6.1^{x} = 6$ (mod 13), kde *x* je celé číslo. Z toho je patrné, že řešení bude nekonečně mnoho, ačkoli pro výpočty v kryptografické praxi postačí samozřejmě jeden. Výsledek lze zapsat třeba jako $k = 5$ (mod 12).

Jak bylo již bylo zmíněno výše, pro výpočet diskrétního logaritmu neexistuje žádný algoritmus, který by dokázal nalézt jeho řešení v polynomiálním čase. Nejjednodušší přístup k řešení problému je postupně zvětšovat hodnotu *k* při umocňování dokud nebude nalezeno požadované *g*. Toto prosté násobení vyžaduje čas, který je závislý lineárně na počtu prvků v cyklické skupině a exponenciálně na rostoucím řádu jednotlivých prvků.

Metoda prostého násobení však není jediná, která lze použít k řešení diskrétního logaritmu. V praxi existuje hned několik různých metod často inspirovaných řešením problému faktorizace, které časovou náročnost výpočtu snižují, žádná ale neřeší problém v polynomiálním čase. Některé z nich budou v krátkosti představeny později.

### 3.4.1 Cyklická grupa G

Grupa je matematickým pojmem definujícím skupinu čísel splňujících jisté vlastnosti. Existují různé druhy grup, jako například aditivní grupy bodů rovinné eliptické křivky nebo multiplikativní grupy. Zde bude text omezen pouze na multiplikativní cyklické grupy, využití eliptických křivek ke stejným účelům je již nad rámec tohoto textu.

Mluví-li se o cyklické multiplikativní grupě G, pak jde o grupu, jejíž prvky jsou tvořeny jedním základním prvkem, generátorem *b*. Všechny prvky grupy daného řádu jsou pak mocninou generátoru . Například může jít o grupu *n*-tých komplexních odmocnin čísla 1. Cyklická multiplikativní grupa však může mít i více než jeden generátor. Taková situace vzniká třeba při operaci modulo *n* v oboru celých čísel [10]. Například operací modulo 9 v multiplikativní grupě s generátorem 5, lze nalézt další generátory 1, 5, 7, 8, 4 a 2.

Takovou skupinu lze z hlediska diskrétního logaritmu definovat jako $log_b : G \rightarrow \mathbf{Z}_n$, kde $\mathbf{Z}_n$, kde $\mathbf{Z}_n$ představuje okruh celých čísel modulo *n*, resp. cyklickou podgrupu řádu *n* jednoho z generátorů grupy G. Pokud řád grupy *n*, je prvočíslem, pak cyklická multiplikativní grupa G má právě *n*-1 generátorů (jde o řadu 1, 2, .., *n*-1). Každý tento generátor vytváří vlastní podgrupu původní grupy, která je opět cyklická a platí pro ni stejná pravidla jako pro nadřazenou grupu. Tabulka 1 je ukázkou grupy řádu 7 a jejích podgrup.

Z pohledu diskrétního logaritmu jsou tyto grupy rovnocenné, pro užití v kryptografii ale z bezpečnostních důvodů nelze využít libovolné grupy G. Využít lze pouze podgrup splňujících DDH podmínky (zkratka Decisional Diffie-Hellman assumption). Tyto podmínky jsou pro multiplikativní cyklické grupy dvě, přičemž platit musí alespoň jedna z nich [11]. Zjednodušeně lze použít grupy, pro které platí že:

1) V *k*-té podgrupě grupy řádu *n* je (*n* - 1) / *k* je rovněž velké prvočíslo, *k* je zbytek dělení modulo *n* v původní grupě.
2) Cyklická grupa je řádu (*p* - 1)(*q* - 1). *p* a *q* jsou bezpečná prvočísla, tj. prvočísla, pro která platí, že *p*=2*x*+1, kde *x* je rovněž prvočíslo. (například 5, 7, 11, 23, 47, 59,…)

*Tabulka 1: Multiplikativní grupa sedmého řádu a její podgrupy*

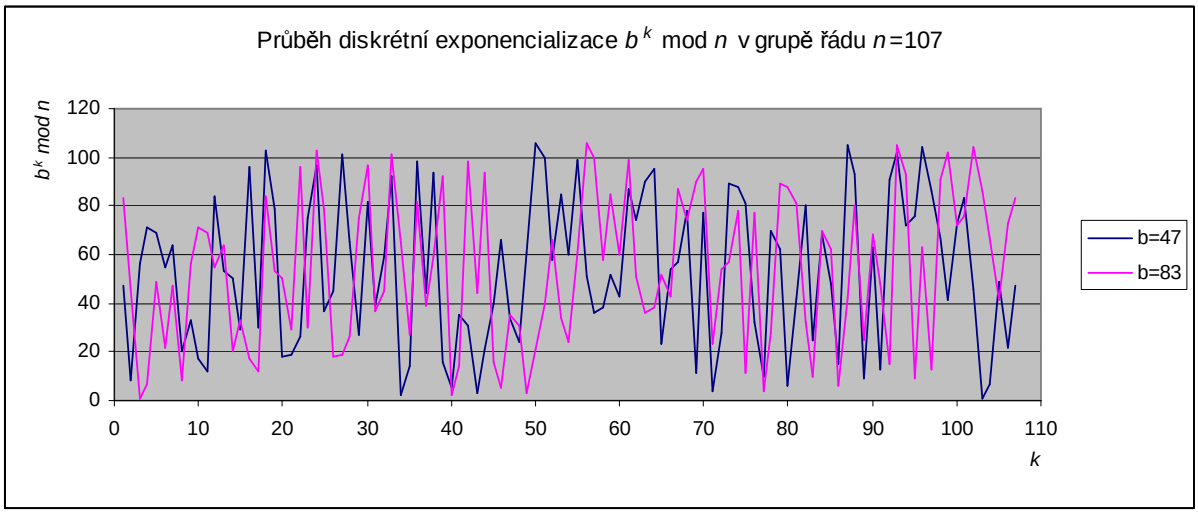
| Řád grupy 7 (modulo 7) | | Mocnina | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Generátor | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 |
| | 3 | 1 | 3 | 2 | 6 | 4 | 5 | 1 | 3 | 2 | 6 | 4 | 5 | 1 | 3 |
| | 4 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 |
| | 5 | 1 | 5 | 4 | 6 | 2 | 3 | 1 | 5 | 4 | 6 | 2 | 3 | 1 | 5 |
| | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 |
| | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 9 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 |
| | 10 | 1 | 3 | 2 | 6 | 4 | 5 | 1 | 3 | 2 | 6 | 4 | 5 | 1 | 3 |
| | 11 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 | 4 |
| | 12 | 1 | 5 | 4 | 6 | 2 | 3 | 1 | 5 | 4 | 6 | 2 | 3 | 1 | 5 |
| | 13 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 | 1 | 6 |
| | 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 3.4.2 Vlastnosti diskrétního logaritmu

Diskrétní logaritmus jako jeden z matematických aparátů využívaných v kryptografii, tak jak je popsán v kapitole 3.4, odvozuje svoje vlastnosti zejména z grup, v nichž se jeho hodnota určuje. V zásadě nezáleží na tom, zda se výpočet provádí v aditivní grupě eliptických křivek nebo multiplikativní grupě modulo *n*, jelikož výsledky diskrétního mocnění se v závislosti na použitém exponentu (mocniteli) *k*, jeví jako náhodný šum. Takový průběh pro malou multiplikativní cyklickou grupu, znázorňuje obrázek 3. V praxi se využívá velkých grup, průběh by tedy měl delší periodu a jeho volatilita by byla vyšší.

Rozdíly mezi grupami se objevují až při bližším pohledu. Bude-li pozornost zaměřena na dobu potřebnou k výpočtu diskrétního logaritmu, pak pro výpočet v aditivní grupě bude

třeba vždy exponenciálně rostoucího času v závislosti na velikosti čísla, jehož diskrétní logaritmus se počítá. Pro výpočet v multiplikativních grupách existují algoritmy, které tento čas zkracují a závislost se stává „pouze“ subexponenciální [12]. I tato závislost však dovoluje považovat multiplikativní grupy za dostatečně vhodné, protože při délkách klíče kolem 1024 bitů (128 cifer dekadického čísla), což se v asymetrické kryptografii v současnosti považuje za minimální standard, je doba potřebná pro výpočet diskrétního logaritmu stále příliš dlouhá, než aby bylo i při rostoucí výpočetní síle možné výpočet provést. Využití eliptických křivek tedy pouze umožňuje použití kratších klíčů, jinak ale výpočet v zásadě neovlivňuje.




*Obr. 3: Průběh hodnot diskrétního mocnění $b^k$ mod n v multipl. grupě řádu n=107*

### 3.4.3 Metoda „baby-step giant-step“

Metoda „baby-step giant-step“ je jednou z metod pro výpočet diskrétního logaritmu. Jde o úpravu metody prostého násobení. Jejím autorem je Daniel Shanks, který ji vymyslel původně jako metodu pro řazení čísel do polí kvadratických čísel. Metoda je ovšem velmi variabilní a ve výpočtech ve skupinách našla velké uplatnění. [13]

Metoda postupuje následujícím způsobem. Určí hodnotu $m = \lfloor n^{1/2} \rfloor$ a pro každou hodnotu $g \in \langle b \rangle$ je $k=log_b\, g$ menší než n, takže lze psát $x + y \times m$ kde $x$ a $y$ je vždy menší než $m$ a větší než 0. Z rovnosti $g = b^{\log_b g} = b^{x+y\times m}$ lze získat pro některé hodnoty $x$ a $y$ rovnost $g \times b^{-ym} = b^x$ . Z toho plyne, že v zápisech $\left(1, b, b^2, ..., b^{m-2}, b^{m-1}\right)$ a $\left(g, gb^{-m}, gb^{-2m}, ..., gb^{-(m-2)m}, gb^{-(m-1)m}\right)$ existují maximálně dvě kolize, tj. $\left(b^{x_0}, gb^{-y_0 m}\right)$ a $\left(b^{x_1}, gb^{-y_1 m}\right)$ takové, že $b^{x_0} = gb^{-y_0 m}$ a $b^{x_1} = gb^{-y_1 m}$ . Hodnota $k$ se získá použitím dvojice s nejmenším $y_i$ a $k = x_i + y_i m$ [14].

Algoritmus je poměrně náročný na paměť, kterou ze redukovat volbou menší hodnoty *m*, což ale prodlouží čas potřebný pro výpočet. Výhodnější je proto použít například některou z variant Pollardova rho algoritmu, která má podobnou časovou náročnost, ale menší náročnost paměťovou.

### 3.4.4 Algoritmus indexového výpočtu

V originálním názvu „index calculus algorithm“ je v současnosti pravděpodobně nejvíce využívaným algoritmem pro výpočet diskrétního logaritmu čísla z cyklické skupiny $\mathbf{Z}_n$, kde *n* je prvočíslem. Tvar algoritmu je opět $b^k=g \pmod n$. Algoritmus vyžaduje zvolit základnu dělitelů z této skupiny, zpravidla tak, že prvním číslem skupiny je -1 a postupně *r* prvočísel následujících po sobě ve skupině. Volba velikosti této skupiny je velmi důležitá z hlediska efektivity algoritmu. Malá skupina je výhodná z hlediska výpočetní efektivity, velká skupina je naopak nutná pro hledání řešení ve velké skupině. Při praktické implementaci se tak zpravidla volí kompromis.

Algoritmus jako takový se provádí ve třech fázích. V první a druhé fázi vyžaduje pouze hodnotu generátoru *b* a hodnotu celočíselného dělitele *n*. Z nich počítá hodnoty diskrétních logaritmů pro základnu malých prvočíselných dělitelů *r*. Třetí fáze pak počítá hodnotu diskrétního logaritmu požadovaného čísla *g* v mezích diskrétních logaritmů základny dělitelů *r*.

Přesný popis algoritmu přesahuje rámec tohoto textu, bližší informace lze nalézt např. v [15]. Důležitým poznatkem však je, že první a třetí fázi algoritmu lze zpracovávat v paralelních systémech a navíc současně. Druhá fáze je ovšem výpočetně poměrně složitá a není ji možné vykonávat paralelně na více počítačích, k jejímu výpočtu se tedy zpravidla používá superpočítačů s velkým výpočetním výkonem.

### 3.4.5 Pollardův rho algoritmus pro logaritmy

Pollardův rho algoritmus pro logaritmy vychází z Pollardova algoritmu pro faktorizaci čísel. Vyznačuje se výpočetní složitostí a časovou náročností obdobnou jako algoritmus „Baby-step giant-step“ ale na rozdíl od něj nepotřebuje tolik paměťového místa pro mezivýsledky, je proto používán výrazně častěji. Vzhledem k jeho relativní jednoduchosti lze snadno programově implementovat, bude proto popsán podrobněji.

Cílem metody je spočítat takové *k*, pro které platí $b^k=g \pmod n$, kde *g* je členem cyklické skupiny *G* s generátorem *b*. Skupina G se rozdělí na 3 zhruba stejně velké podskupiny $S_1$, $S_2$, $S_3$, pro které musí platit některé podmínky (například číslo 1 nesmí patřit do $S_2$). Definuje se posloupnost prvků $x_0$, $x_1$, $x_2$… tak, že $x_0=1$ a

$$x_{i+1} = f(x_i) \overset{\text{def}}{=} \begin{cases} g.x_i, & \text{pokud } x_i \in S_1 \\ x_i^2, & \text{pokud } x_i \in S_2 \\ b.x_i, & \text{pokud } x_i \in S_3 \end{cases} \quad \text{pro } i \geq 0. \qquad (4.1)$$

Tato posloupnost prvků pak definuje dvě skupiny celých čísel $A_0$, $A_1$, $A_2$,… a $B_0$, $B_1$, $B_2$,… splňující podmínky $x_i = b^{A_i} g^{B_i}$ pro $i \geq 0 : A_0 = 0,\ B_0 = 0$, a pro $i \geq 0$:

$$A_{i+1} = \begin{cases} A_i, & \text{pokud } x_i \in S_1 \\ 2.A_i \bmod n, & \text{pokud } x_i \in S_2 \\ A_i + 1 \bmod n, & \text{pokud } x_i \in S_3 \end{cases} \qquad (4.2)$$

a

$$B_{i+1} = \begin{cases} B_i + 1 \bmod n, & \text{pokud } x_i \in S_1 \\ 2.B_i \bmod n, & \text{pokud } x_i \in S_2 \\ B_i, & \text{pokud } x_i \in S_3 \end{cases}. \qquad (4.3)$$

S použitím Floydova algoritmu lze nalézt prvky $x_i$ a $x_{2i}$ takové, že $x_i = x_{2i}$. Z toho $b^{A_i} g^{B_i} = b^{A_{2i}} g^{B_{2i}}$ a odtud $g^{B_i - B_{2i}} = b^{A_{2i} - A_i}$. Zlogaritmováním této poslední rovnosti logaritmem o základu *b* vznikne rovnice

$$(B_i - B_{2i}) \log_b g \equiv (A_{2i} - A_i) \pmod n.$$

Získáním $B_i \neq B_{2i} \pmod n$ (pravděpodobnost, že se prvky budou rovnat je zanedbatelná) lze z tuto rovnici efektivně řešit k určení $log_b\, g$ [16] [17]

Vypočetše vstupními hodnotami generátoru cyklické skupiny *b*, jejím prvočíselným řádem *n* a prvkem $g \in G$ probíhá v následujících krocích:

1) Nastaví $x_0=1$, $A_0=1$, $B_0=1$
2) Pro *i* = 1, 2, …
   a) S použitím hodnot $x_{i-1}$, $A_{i-1}$, $B_{i-1}$ a $x_{2i-1}$, $A_{2i-1}$, $B_{2i-1}$ spočtených v předešlém kroku urči hodnoty $x_i$, $A_i$, $B_i$ a $x_{2i}$, $A_{2i}$, $B_{2i}$ s použitím vztahů (4.1), (4.2) a (4.3).
   b) Porovná $x_i$ a $x_{2i}$ a
      - do pomocné proměnné *r* vloží ($B_i$ a $B_{2i}$) mod *n*
      - pokud *r* = 0, skončí s chybovým hlášením (lze ošetřit novými výpočty s jiným rozdělením skupiny G do částí)
      - pokud r $\neq$ 0, pak $k = r^{-1}(A_{2i} - A_i) \bmod n$

# 4 KRYPTOSYSTÉMY ZALOŽENÉ NA PROBLÉMU DISKRÉTNÍHO LOGARITMU

Jak již bylo řečeno v předchozím textu, úplných matematických problémů se hojně využívá v asymetrické kryptografii pro návrh šifer, respektive jednosměrných funkcí šifrovacích systémů. Jednosměrná funkce je zjednodušeně taková funkce, kterou lze snadno vyřešit (v polynomiálním čase) zatímco určit z jejího výsledku původní vstupy funkce je velice obtížné (řešení lze najít v exponenciálním nebo dokonce faktoriálním čase). Předpokládá se, že v současné době pravděpodobně neexistuje efektivní algoritmus k jejich řešení. Slovo pravděpodobně je zde zcela namístě, protože finanční hodnota takového algoritmu je obrovská a její existence by nebyla pravděpodobně veřejně publikována.

Kryptosystémy založené na NP problémech lze tedy při dodržení všech podmínek kladených na vstupní hodnoty a délky klíčů považovat za dostatečně bezpečné. Tento text je zaměřen na problematiku diskrétních logaritmů a kryptosystémů na nich založených, proto v dalším textu budou alespoň ty nejznámější a nejvýznamnější podrobně představeny.

## 4.1 Diffie-Hellmanův protokol

Diffie-Hellmanův protokol je protokolem asymetrické kryptografie určeným pro ustavení tajného klíče mezi komunikujícími uživateli po veřejných nezabezpečených sítích bez toho, aby uživatelé mezi sebou před zahájením komunikace sdíleli jakékoli tajemství. Existuje v mnoha mírných modifikacích, jeho bezpečnost je však vždy založena na problému diskrétního logaritmu, který není při dodržení podmínek definovaných v kapitole 3.4 řešitelný v rozumném čase. Jeho autoři Diffie a Hellman, podle kterých se protokol jmenuje, publikovali metodu v roce 1976 a dali tak základ celé asymetrické kryptografii. Ačkoli metodu zveřejnili o dva roky dříve, než byla publikována metoda RSA, nikdy nedosáhla takové popularity ani rozšíření. Byla však základem a inspirací mnoha dalším metodám, které z ní přímo či nepřímo vycházejí a stále existují aplikace, nabízející volitelně její použití.

Metoda pro komunikaci mezi dvěma stranami, Alicí a Bobem, probíhá následujícím způsobem:

1) Jedna z komunikujících stran (Alice) zvolí velké prvočíslo *n* jako řád multiplikativní grupy $\mathbf{Z}_n^*$ a generátor *b* z této grupy, platí $1 \leq b \leq n-2$. Hodnota n musí být taková, aby

hodnota *n-1* měla alespoň jeden velký prvočíselný dělitel nebo *n-1= 2p*, kde *p* je prvočíslo. Hodnoty *n* a *b* jsou veřejné.

2) Alice si z vybrané grupy zvolí soukromý klíč *x* ($1 \leq x \leq n-2$) , vypočte

   $A : b^x \bmod n$

   a vypočtenou hodnotu odešle nezabezpečeným způsobem Bobovi.

3) Bob si z vybrané grupy zvolí svůj soukromý klíč *y* ( $1 \leq y \leq n-2$ ), vypočte

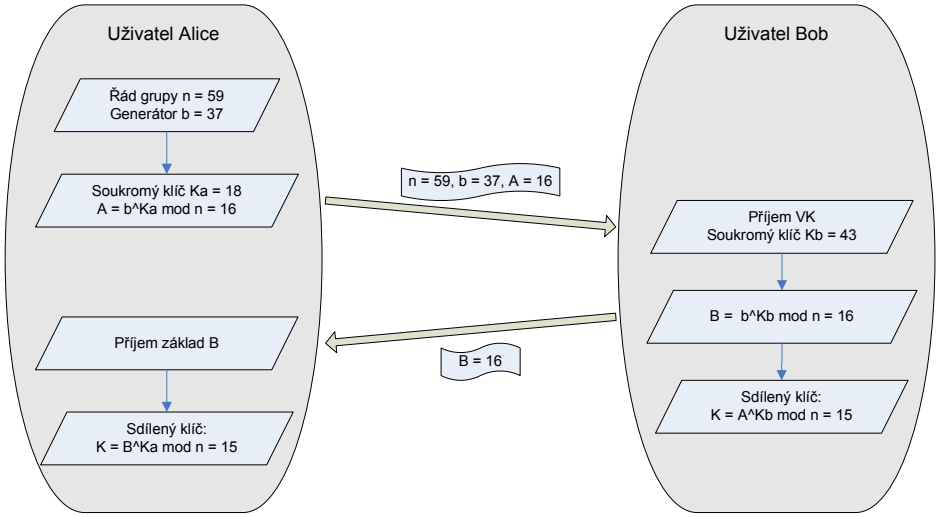
   $B : b^y \bmod n$

   a vypočtenou hodnotu odešle nezabezpečeným způsobem Alici.

4) Alice od Boba přijme zprávu $B=b^y$ a vypočte sdílený klíč $K = \left(b^y\right)^x \bmod n$

5) Bob od Alice přijme zprávu $A=b^x$ a vypočte sdílený klíč $K = \left(b^x\right)^y \bmod n$

Grafické znázornění komunikace poskytuje Obr. 4. Oba uživatelé si tedy dokážou vypočítat stejný sdílený klíč, jehož hodnotu nedokáže určit nikdo jiný. K tomu aby to dokázal, musel by z přenášené komunikace určit hodnoty *x* a *y*, tedy vyřešit diskrétní logaritmus.




*Obr. 4: Ustavení klíče pomocí Diffie-Helmanova protokolu*

### 4.1.1 Bezpečnost Diffie-Hellmanova protokolu

Z dříve řečeného je zjevné, že nikdo není schopen pasivním odposloucháváním přenosového kanálu určit hodnotu klíče. Slabinou metody je však to, že žádným způsobem neověřuje autentičnost a autorizaci zpráv přenášených v bodech 2 a 3. Tím umožňuje jednoduchý útok „man in the middle," kdy se útočník vloží mezi komunikující strany, dohodne si vlastní klíče s

oběma stranami a s jejich pomocí si dešifruje všechny zprávy, aniž by komunikující strany tušily jakýkoli problém.

Řešením problému může být, že obě strany si nechají hodnoty $b^x$ a $b^y$ (komunikační „podklíče“) podepsat certifikační autoritou a získají certifikáty. Při komunikaci si pak strany vzájemně vymění svoje certifikáty, ověří jejich pravost a platnost a z hodnoty v certifikátu určí sdílený klíč [17]. Použití ověřených podklíčů však znamená, že nedochází k jejich častým změnám a roste tak riziko jejich prozrazení. To naštěstí není příliš časté a jelikož pro výpočet klíče útočník musí mít hodnoty podklíčů obou stran, riziko je minimální.

Častým případem je stav, kdy uživatel s neautorizovaným heslem inicializuje spojení s uživatelem, který disponuje platným certifikátem. Toho lze využít například při e-mailové komunikaci. Uživatel odesílající zprávu si z ověřeného certifikátu vezme veřejný podklíč příjemce zprávy, vypočte sdílený klíč, zašifruje zprávu a spolu s vlastním veřejným podklíčem ji odešle příjemci. Má tak jistotu, že zprávu si přečte pouze vlastník certifikátu, který zná tajnou hodnotu svého klíče. Nedochází tak k žádné předchozí komunikaci mezi oběma stranami, odesílatel zajistil, že si zprávu přečte pouze zamýšlený příjemce, příjemce však nemá možnost ověřit, kdo skutečně zprávu vytvořil [18].

## 4.2 ElGamalův algoritmus

ElGamalův algoritmus byl poprvé představen v roce 1984 panem Tather Elgamalem. Nikdy nebyl patentován, protože vlastníci Diffie-Hellmanova algoritmu jej pokládali za srovnatelný se svým systémem a tedy za chráněný jejich vlastním patentem.

Jde o asymetrický kryptosystém jehož bezpečnost je založena na problému diskrétního logaritmu. Lze jej využít buď k šifrování nebo k podepisování zpráv, ale Vzhledem k tomu, že k jejich představení došlo až několik let po vytvoření RSA a D-H protokolu, nedošlo k masivnímu rozšíření ElGamalových algoritmů. Dnes již z bezpečnostního hlediska původní návrhy nelze použít, přesto některé protokoly nebo aplikace nabízejí volbu upravených ElGamalových algoritmů. Jde však pouze o okrajové využití, zejména pro podepisování zpráv je vhodnější využít algoritmu DSA, jehož bezpečnost je na mnohem vyšší úrovni. K masivnímu rozšíření nedošlo také proto, že šifrovaná nebo podepsaná zpráva algoritmem ElGamal má dvojnásobnou délku oproti zprávě původní.

ElGamalův kryptosystém pro šifrování i podepisování začíná téměř stejně, vytvořením veřejného a privátního klíče. Uživatel tedy postupuje následovně:

1) Vygeneruje náhodné velké prvočíslo *n* a generátor *b* multiplikativní grupy $(\mathbf{Z}_n)^x$ celých čísel modulo *n*.
2) Zvolí náhodné celé číslo k, pro které platí $1 \le k \le n-2$ a vypočte $b^k \bmod n$.
3) Veřejný klíč je pak ($n$, $b$, $b^k$), soukromý klíč potom $k$.

S praktickými hodnotami pak může vypadat situace třeba takto:
Uživatel si zvolí cyklickou grupu řádu $n = 61$ a generátor z této grupy $b = 13$. Jako soukromý klíč si určí hodnotu $k = 44$. Podle uvedeného vztahu vypočte poslední parametr veřejného klíče $Y = b^k \bmod n = 47$. Veřejný klíč je tedy tvořen uspořádanou trojicí $VK$ = [$n$=61, $b$=13, $Y$=47] a soukromý klíč je roven SK = [$k$ = 44].

V popisu se uvádí výběr prvků z multiplikativní grupy, je však možné stejně výhodně použít některou aditivní grupu bodů eliptické rovinné křivky, k přechodu na jejich využití zde ale není žádný výraznější důvod, proto se nepoužívají. Z hlediska bezpečnosti je nutné, aby multiplikativní cyklická skupina splňovala všechny požadavky tak, jak jsou uvedeny v kapitole 3.4.1. Její parametr *n*, tedy řád grupy, je dnes již minimálně 1024 bitů dlouhé číslo [19].

### 4.2.1 ElGamalův kryptosystém pro šifrování zpráv

ElGamalův kryptosystém využívá pro svoji funkci dvojici veřejného a privátního klíče tak, jak jsou popsány výše. Šifrovaná komunikace pak probíhá následovně:

1) Komunikující strana si obstará veřejný klíč protistrany z důvěryhodného zdroje
2) Rozdělí zprávu do více úseků, označených M, pro které platí, že číselná velikost M je v rozsahu $1 \le M \le n-1$
3) Zvolí si náhodné číslo $x$ z intervalu $1 \le x \le n-2$ tak, aby bylo nesoudělné s *n-1*
4) Vypočte $\gamma = b^x \bmod n$ a $\delta = M.(b^k)^x \bmod n$
5) Odešle šifrovaný text $c = (\gamma, \delta)$

Příjemce (vlastník privátního klíče) přijme šifrovaný text $c$:

1) Použije privátní klíč $k$ pro výpočet $A = \gamma^{n-1-k} \bmod n$ (poznámka: $\gamma^{n-1-k} = \gamma^{-k} = b^{-kx}$)
2) Vypočte původní hodnotu $M$ jako $M = A.\delta \bmod n$.

Důkaz, že obnovená zpráva je shodná s původní zprávou je vidět z následující rovnice: $\gamma^{-k}.\delta \equiv b^{-kx}Mb^{kx} \equiv M \pmod{n}$ [17].

Výhoda ElGamalova kryptosystému pro šifrování oproti například RSA je v tom, že do šifrovacího procesu vnáší jistou náhodnost volbou parametru *x*. Na druhou stranu je však nutné tento parametr v průběhu komunikace neustále náhodně měnit a po použití důkladně zničit. V případě, že by se parametr *x* neměnil, začaly by se v obraze při šifrování stejných částí textu (například slov, písmen,…) objevovat shodné průběhy šifrovaných dat a bylo by možné šifru snadno rozlomit například i důkladným statistickým útokem. Nutnost zničení hodnoty *x* je na druhou stranu dána tím, že pomocí této hodnoty by bylo možné šifrované zprávy dopočítat v případě, že by byl k dispozici jeden nešifrovaný text, jemu odpovídající šifrovaný text a další texty šifrované pomocí stejného parametru *x* (Diffie-Hellman parametr *x* označovali jako dočasný privátní klíč odesílatele). Systém lze považovat za bezpečný, pouze však do doby, kdy je jako bezpečný považován Diffie-Hellmanův protokol.

Následující příklad je praktickou ukázkou funkce algoritmu.

*Tvorba klíče*: Uživatel Alice si zvolí řád grupy, například $\mathbf{Z}_{587}$ a generátor z této grupy, libovolně třeba $b = 3$. Současně si zvolí soukromý klíč $k = 263$ a vypočte:

$$b^k \bmod n = 3^{263} \bmod 587 = 416$$

Veřejný klíč Alice je tedy (527, 3, 416).

*Šifrování*: Uživatel Bob si obstará veřejný klíč Alice. Chce šifrovaně odeslat zprávu „ahoj“. Zprávu tedy vyjádří pomocí ASCII tabulky v číselném tvaru: „97 104 111 106.“ Zprávu musí rozdělit do 4 částí, aby byla splněna podmínka $1 \le M \le n-1$. Šifruje první část zprávy M=97. Zvolí si libovolné celé číslo $x = 315$ z intervalu $1 \le x \le n-2$ a vypočte

$$\gamma = 3^{315} \bmod 587 = 280$$

a

$$\delta = 97.416^{315} \bmod 587 = 467$$

Bob odešle šifrovanou zprávu $c_1 = (280, 467)$.

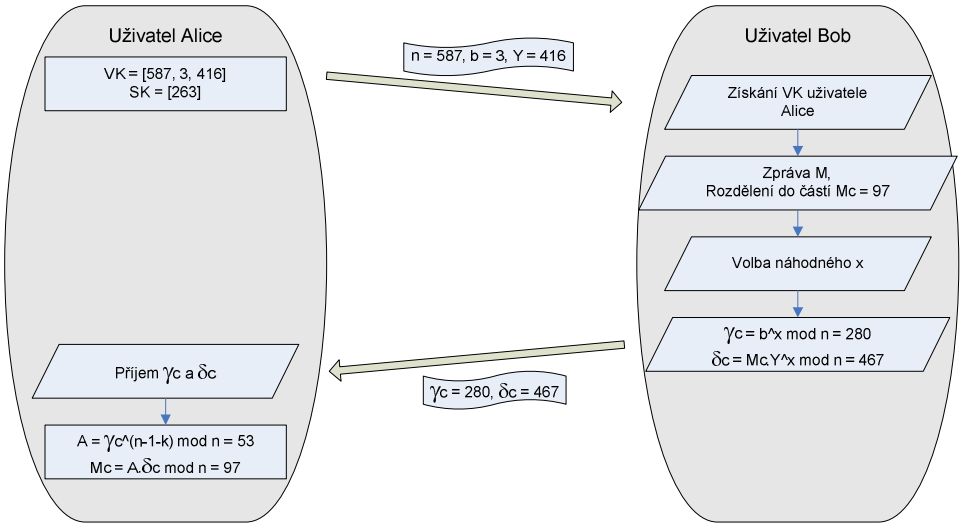
*Dešifrování*: Alice přijme zprávu $c_1$ a vypočte

$$A = 280^{587-1-263} \bmod 587 = 53$$

a

$$M = 53.467 \bmod 587 = 97$$

Alice tedy získá původní zprávu, kterou Bob zaslal. Stejným postupem by Bob zašifroval další prvky zprávy (pro každý prvek ale musí použít jiné *x*) a Alice dešifrovala.




*Obr. 5: Přenos šifrované zprávy protokolem ElGamal*

### 4.2.2 ElGamalův systém pro podepisování zpráv

Zatímco ElGamalův kryptosystém pro šifrování zpráv má i v současné době jistou úlohu jako alternativa k RSA, založená na jiném matematickém problému, ElGamalovo podpisové schéma již pro současné použití není v ničem výhodné, zejména díky slabinám, které byly postupem času objeveny a odstraněny v systému DSA, který z ElGamalova schématu vychází. Přesto však svou úlohu ve své době sehrál a posunul vývoj kupředu, proto je vhodné jej také představit.

Jde v podstatě o jistou variaci schématu ElGamalova kryptosystému, kde se opět vyžívá matematických vlastností diskrétního logaritmu. Uživatel před začátkem komunikace vytvoří dvojici klíčů, z nichž jeden mu poslouží pro vytvoření popisu a druhý bude sloužit ostatním uživatelům pro ověření pravosti tohoto podpisu [21]. Kromě této dvojice klíčů bude uživatel potřebovat také hash (pojem bude rozebrán později) podepisované zprávy *h*(M). existuje několik postupů výpočtu, zde je jeden z nich. [22]

Uživatel tedy nejprve vytvoří dvojici klíčů:

1) Vygeneruje náhodné velké prvočíslo *n* a generátor *b* multiplikativní grupy $(\mathbf{Z}_n)^x$ celých čísel modulo *n*.
2) Zvolí náhodné celé číslo k, pro které platí $1 \le k \le n-2$ a vypočte $y = b^k \bmod n$.
3) Veřejný klíč je pak (*n*, *b*, *y*), soukromý klíč potom *k*.

Uživatelka Alice chce podepsat svoji zprávu M, postupuje tedy následujícím způsobem:

1) Zvolí si náhodné, tajné, celé číslo x, pro které platí $1 \le x \le n-2$ a je nesoudělné beze zbytku s *n-1*

2) Vypočte $r = b^x \bmod n$
3) Vypočte $s = k.r + (x.h(M)) \bmod (n-1)$
4) Podpis zprávy M je dvojice (*r*, *s*)

Pokud se jiný uživatel rozhodne ověřit si pravost zprávy, postupuje následujícím způsobem:

1) Z důvěryhodného zdroje si obstará veřejný klíč uživatelky Alice (*n*, *b*, *y*)
2) Ověří, že $1 \le r \le n-1$, pokud toto není splněno, podpis automaticky označí za neplatný
3) Vypočte *h* (M) a $v_1 = y^r r^{h(m)} \bmod n$
4) Vypočte $v_2 = b^s \bmod n$
5) Za předpokladu, že $v_1 = v_2$, podpis přijme. V opačném případě označí zprávu za falešnou. Podpis samozřejmě nebude odpovídat ani v případě, že při přenosu došlo k chybě a záměně bitu kterékoli části zprávy nebo klíče.

Stejně jako v případě ElGamalova kryptosystému pro šifrování zpráv, tak i v tomto případě je nutné pro každou podepisovanou zprávu volit odlišné a náhodné číslo *x*, aby se udržela základní bezpečnost podpisu. Obdobně kvůli autentičnosti zprávy je vždy nutné podepisovat její hash, protože existují matematické metody, jak v opačném případě bezpečnost napadnout. Již výše bylo zmíněno, že tento způsob podepisování se již nepoužívá, avšak pro pochopení podstaty podepisování zpráv je metoda velmi vhodnou.

Zde je příklad výpočtu ElGamalova podpisu a jeho ověření.

*Tvorba klíče:* Vzhledem k tomu, že klíč ElGamalova algoritmu se vytváří stejně jak pro účely šifrování, vypočte se veřejný klíč například 101, 3, 28 a soukromý klíč *k* = 19.

*Podepsání zprávy*: Zprávu, kterou bude podepisovat, je rovna M = 778. Její hash (viz dále) vypočte jako *h(M)* = 165. Nyní si zvolí libovolné tajné číslo z grupy veřejného klíče, třeba *x* = 96 a vypočítá obě části podpisu:

$$r = 3^{96} \bmod 101 = 5$$

a

$$s = 19.5 + (96.167) \bmod (n-1) = 127$$

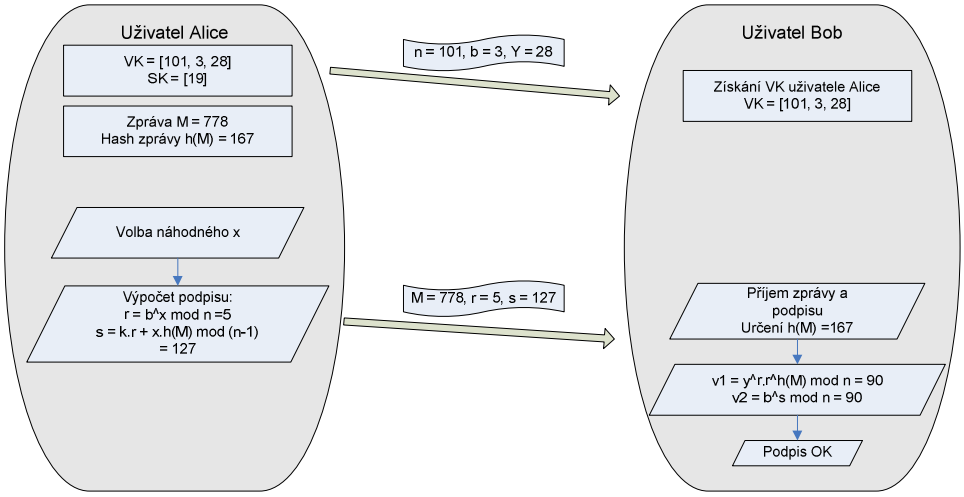
Spolu s odesílanou zprávou M tedy Alice odešle ještě podpis (r = 5, s = 127).

*Ověření podpisu*: Uživatel Bob přijme zprávu od Alice spolu s jejím podpisem. Aby mohl zprávu ověřit, musí znát ještě veřejný klíč patřící Alici. Ten si obstará z důvěryhodného zdroje a může začít s ověřováním podpisu. Nejdříve vypočte hash přijaté zprávy *h(M)* = 165 a následně parametry $v_1$ a $v_2$:

$$v_1 = 28^5 5^{165} \bmod 101 = 90 ,$$

$$v_2 = 3^{127} \bmod 101 = 90$$

Z rovnosti parametrů odvodí, že autorem zprávy je opravdu uživatelka Alice a zprávu přijme.




Obr. 6: Vytvoření a ověření podpisu mezi dvěma uživateli

### 4.2.3 Bezpečnost ElGamalova kryptosystému

ElGamalův kryptosystém byl ve svojí době na velice dobré bezpečnostní úrovni a i v dnešních podmínkách disponuje vlastnostmi, které jsou z hlediska bezpečnosti velice výhodné. Jako celek již však není považován za zcela bezpečný, zejména pak použití podpisového schématu se nedoporučuje. Šifrovací schéma stále lze bezpečně využívat, ale vzhledem k prodloužení přenášené zprávy na dvojnásobek se k praktickému nasazení algoritmu nepřistupuje.

Zmíněnou bezpečnostní výhodou oproti jiným kryptosystémům je použití náhodného parametru *x* při výpočtu šifrovaného bloku zprávy. Jelikož se pro každý blok zprávy parametr mění, mění se i parametry důležité z hlediska implementace algoritmu, zejména čas potřebný pro výpočet. Jak bude popsáno v kapitole věnované postranním kanálům, je tato vlastnost zcela zásadní a algoritmy jako RSA, které tuto vlastnost nemají, obsahují velkou bezpečnostní slabinu (musí se provádět dodatečné úpravy, aby systém byl považován za bezpečný).

Na druhou stranu, pokud by se parametr v konkrétní implementaci neměnil, dokázal by oslabit bezpečnost algoritmu na naprosté minimum a v nejhorším případě by k prolomení šifry stačil statistický útok na dostatečné množství šifrovaných textů s alespoň částečně známým obsahem.

## 4.3 Hashovací funkce založená na diskrétním logaritmu

### 4.3.1 Hash zprávy

Než bude popsána konkrétní hashovací funkce vycházející z diskrétního logaritmu, bude účelné popsat, co vlastně hashovací funkce je a k čemu slouží.

Hashovací funkce je takzvaný jednosměrný otisk zprávy, tj. jedinečný kód, vzniklý z původní zprávy použitím speciální funkce. Ta převede zprávu z množiny všech možných zpráv M do množiny všech možných hashů H. Jelikož množina M je mnohonásobně větší než množina H, existuje mnoho zpráv, k nimž náleží stejný hash. Pokud je však hashovací funkce správně navržena, není možné v reálném čase cíleně najít zprávu, které náleží stejný hash, jako původně zvolené zprávě.

Mezi nejznámější a nejpoužívanější hashovací funkce patří například MD5, SHA1, SHA2 atd. Tyto funkce jsou všeobecně známy a každý je může využít k výpočtu hashe zprávy. Důležité rovněž je, že získaného hashe není možné žádným způsobem obnovit zprávu, ze které byl vytvořen. Hash není závislý na délce původní zprávy, jeho délka je konstantní podle použité funkce a některé části dat jsou tedy ztraceny.

Pokud se podaří najít jinou zprávu než původní, která má stejný hash, dochází ke kolizi. Pokud se takové zprávy daří nalézat cíleně, pak se hashovací funkce stává nepoužitelnou. To je v současnosti případ funkce MD5, jejíž bezpečnost již byla prolomena a kolize lze nalézat v řádu několika minut.

### 4.3.2 Shamirova hashovací funkce

Hashovací funkce založená na výpočtu diskrétního logaritmu byla před lety představena jedním z tvůrců kryptosystému RSA, Adi Shamirem. Jde o jednoduchý návrh hashovací funkce, jejíž odolnost vůči kolizím je dána obtížností řešení diskrétního logaritmu a obtížností faktorizace součinu velkých prvočísel. Výhodou této funkce je zejména její jednoduchost a prokazatelná odolnost proti kolizím.

Funkci samotnou pak Adi Shamir definoval následujícím způsobem:

Nechť $n = p.q$ kde čísla $p$ a $q$ jsou velká prvočísla, takže faktorizace čísla $n$ je časově extrémně náročná.

Nechť $b$ je prvkem grupy $\mathbf{Z}_n^*$ (tedy například prvkem řádu $\lambda(n)=\mathrm{lcm}(p-1, q-1)$)

Předpokládá se, že *n* a *b* jsou daná a veřejně známá čísla, *p* a *q* jsou tajná. Potom jestliže *x* je vstupní hodnota, která má být hashována, vyjádřená jako nezáporné celé číslo libovolné délky (i výrazně větší než je délka *n*), je hashovací funkce definována jako

$$\text{hash}(x) = b^x \bmod n .$$

Předpokládaná odolnost proti kolizím je tedy vyjádřena jako obtížnost nalezení $x'$ a $x$ takových, aby platilo

$$\text{hash}(x) = \text{hash}(x').$$

Přímé nalezení takovéto kolize je problematické vzhledem k nutnosti vyřešit diskrétní logaritmus. Z uvedené rovnice také plyne, že $x - x' = k.\ \lambda(n)$ pro libovolné celočíselné $k$ (způsobeno cyklickými vlastnostmi grupy). To naznačuje výskyt kolizí pro mnoho hodnot λ(n). Nalezení takových hodnot λ(n) však předpokládá schopnost provést faktorizaci *n*. Celé odvození těchto tvrzení je možné nalézt v [21].

Tato hashovací funkce založená na diskrétním logaritmu je v mnoha ohledech specifická a odlišuje se od ostatních. Jedno specifikum již bylo zmíněno, je jím velmi jednoduchý návrh, který vzhledem ke stejnému principu jako výpočet asymetrického šifrování nevyžaduje po kryptosystému speciální funkce (obvody) pro hashování zpráv. Mechanismus je navíc natolik jednoduchý a přehledný, že lze určit obtížnost výpočtu kolizí. Jistou nevýhodou zde je, že vypočtený hash má proměnnou délku závislou na modulárním základu *n*. To vzhledem k tomu, že ostatní hashovací funkce dávají vždy konstantní délku výstupu může představovat problém pro některé aplikace s přesnou strukturou dat. Problém lze řešit například rozdělením výsledných dat do bloků a následně jejich zpracováním operací XOR všech částí dohromady.

Velmi výraznou odlišností oproti jiným hashovacím funkcím je použití individuálních klíčů *p* a *q* na straně uživatele, který hash vytváří. Jejich existence dává funkci mnohem větší bezpečnost oproti ostatním hashovacím funkcím. Teoreticky ale jejich použití dává uživateli možnost počítat kolize hashovací funkce a podvrhnout falešnou zprávu proti vlastní původní zprávě. Vyvstává zde tedy otázka určité důvěry, jejímž řešením do budoucna je například využití služeb důvěryhodné třetí strany (určitá obdoba certifikační autority), která by zajišťovala správu klíčů pro uživatele.

## 4.4 DSA – Digital Signature Algorithm

Standard DSA je standard určený pro generování a ověřování digitálních podpisů zpráv. Vznikl jako nástupce ElGamalova schématu pro podepisování zpráv, ale odstraňuje některé jeho chyby. Někdy se zaměňuje s označením DSS (Digital Signature Standard), což je americký standard digitálního podpisu, který z DSA vznikl v roce 1991. DSS je s DSA v podstatě totožný, proto záměna označení není chybou. Rozdílem je, že DSS specifikuje maximálně 1024bitů dlouhý klíč, DSA takové omezení nemá. V současné praxi se využívá klíčů délek 2048bitů pro podpisy běžných uživatelů a 4096 bitů v certifikačních autoritách.

Podpisové schéma opět využívá diskrétního logaritmu v konečné cyklické grupě $\mathbf{Z}_n^*$ a jeho bezpečnost je závislá na neschopnosti diskrétní logaritmus efektivně řešit. Doposud přesto nebylo prokázáno, že je to dostatečnou podmínkou pro bezpečnost, stejně jako není prokázána bezpečnost RSA digitálního podpisu [23].

Jak pracuje digitální podpis a k čemu je vhodné ho využívat bylo popsáno kapitole 2.2.1 proto je možné přistoupit přímo k mechanismu podepisování. Před podepisováním zprávy je třeba vytvořit dvojici soukromého a veřejného klíče. Jejich tvorba probíhá v následujících krocích [24]:

1) Zvolí prvočíslo *n* takové, že $2^{511+64t} < n < 2^{512+64t}$ $(0 < t < 24)$ a prvočíslo *q*, které je celočíselným dělitelem *n-1* (zpravidla délky 160-512 bitů).
2) Zvolí parametr h takový, že *h* < *n-1* a současně $h^{(n-1)/q} \bmod n > 1$
3) Vypočte generátor *b* jako $b = h^{(n-1)/q} \bmod n$
4) Zvolí náhodné celé číslo k, pro které platí $1 < k < q$
5) Vypočte $y = b^k \bmod n$
6) Veřejný klíč je tvořen kombinací (*n, q, b, y*), privátní klíč je *k*.

Vytvoření podpisu se provádí následujícím způsobem:

1) Zvolí se náhodné celé číslo x, pro které platí $0 < x < n$
2) Vypočte $r = (b^x \bmod n) \bmod q$
3) Vypočte $s = x^{-1}\{h(\mathrm{M}) + kr\} \bmod q$
4) Digitální podpis zprávy M je dvojice (*r*, *s*)

Pro vytvoření otisku (hashe) zprávy *h*(M) využívá DSA algoritmu SHA-1, který vytvoří 160 bitů dlouhou sekvenci dat, kterou je třeba pro potřeby výpočtu převést na číslo. Ověření pravosti podpisu (*r*, *s*) zprávy M se provádí následujícím způsobem:

1) Uživatel získá z důvěryhodného zdroje veřejný klíč (*n, q, b, y*)

2) Ověří, že $0 < r < q$ a $0 < s < q$. Pokud není splněna jedna z podmínek, podpis odmítne
3) Vypočte $w = s^{-1} \bmod q$ a $h(\mathrm{M})$
4) Vypočte $u_1 = w.h(\mathrm{M}) \bmod q$ a $u_2 = r.w \bmod q$
5) Vypočte $v = (b^{u_1} y^{u_2} \bmod n) \bmod q$
6) Podpis přijme v případě, že $v = r$, v opačném případech podpis zprávy odmítne.

Vzhledem k výpočetní náročnosti nebude uveden ani praktický příklad podepsání zprávy, DSA podepisování se provádí výhradně s pomocí výpočetní techniky.

V současnosti se systém DSA stále hojně používá, postupně se však přechází k jeho vylepšené variantě ECDSA, která místo multiplikativních cyklických grup využívá aditivní grupy eliptických křivek. Samozřejmě pro bezpečnou distribuci veřejných klíčů je nutné využívat služeb některé z certifikačních autorit.

## 4.5 Obecná bezpečnost kryptosystémů DL

Bezpečnost kryptosystémů založených na problému diskrétního logaritmu lze rozdělit na dvě skupiny. První skupina byla částečně představena u popisu jednotlivých systémů, jde o problematiku samotného matematického mechanizmu a všech podmínek, které musejí být splněny, aby systém odolal matematickým metodám řešení popsaným v této práci.

Druhou skupinu tvoří bezpečnost z hlediska implementace algoritmu do fyzického zařízení, tzv. kryptografického modulu, který se určitým způsobem projevuje vůči svému okolí a tím z něho unikají informace, které mohou být využity pro kryptoanalýzu.

### 4.5.1 Postranní kanály v kryptografii

Postranní kanály jako kryptografický pojem vyjadřují skupinu informací, kterými se projevuje kryptografický modul vůči svému okolí. Z těchto informací lze často získat důležité pomůcky k prolomení šifry, kterou modul počítá. Jako první s jejich využitím přišel pan Kocher, který využil časových závislostí v modulech pro RSA šifrování jejich časových vlastností a úspěšně prolomil šifru RSA. Od té doby se postranní kanály staly jedním z nejzásadnějších problémů pro tvorbu kryptosystémů.

Pro bezpečnost kryptosystémů DL je nejčastější problém s již zmíněným časovým postranním kanálem. Každý modul, počítající podle známého algoritmu potřebuje určitý

strojový čas. Pokud je tento čas možné změřit, pak je možné usuzovat na parametry, které jsou počítány.

Konkrétně u výpočtu diskrétního mocnění, které se v modulech provádí je z pohledu útočníka následující situace. Je známo číslo, které se umocňuje – generátor grupy, dále číslo kterým se dělí umocněný výsledek – řád grupy. Není znám exponent, kterým se umocňuje – soukromý klíč. Útočník ví, že výpočet zbytku po celočíselném dělení musí probíhat postupně, aby čísla vznikajícím mocnění nebyla neúměrně vysoká. Pokud by byl výpočet prováděn podle algoritmu binárního mocnění jak je popsán v kapitole 5.2.1, šlo by o řadu shodných kroků, kde v prvním kroku zná všechny parametry výpočtu a další kroky užívají výsledek z předchozího kroku. Zda byl bit klíče roven nula nebo jedna by usoudil podle toho, zda modul prováděl výpočet násobení získaných kořenů nebo ne.

K tomu, aby byl schopen útok provést, potřebuje měřit časy, které jednotlivé operace modulu zabírají. K tomu využije dalšího postranního kanálu.Zde má volbu hned z několika možností. Pokud procesor počítá násobení dvou čísel, spotřebovává mnohem více energie než například pro adresaci do své paměti, v odebíraném výkonu tak vznikají pravidelná minima a maxima a čas mezi nimi je čas mezi dílčími kroky výpočtu. Současně se tím mění elektromagnetické spektrum modulu, útočník tedy může sledovat spektrum. Při výpočtech procesor navíc není úplně potichu, ale vydává zvuky jejichž frekvence se mění v závislosti na činnosti modulu.

Díky těmto získaným informacím může útočník přesně usuzovat, co právě procesor dělá a podle počtu jednotlivých obrazů v získaném spektru přesně určí tajný klíč, s jehož pomocí už data dešifruje.

Proti popsanému útoku má určitou výhodu ElGamalův kryptosystém, který při výpočtech jednotlivých částí zprávy volí vždy náhodný parametr $x$ a čas při šifrování se tak náhodně mění. Při výpočtu veřejného klíče ze soukromého však žádná náhodnost nenastává a tvorba klíče se tedy musí řešit v dostatečně zabezpečeném modulu.

Z popsaného je vidět, že ačkoli diskrétní logaritmus nelze efektivně vyřešit, implementace diskrétního mocnění však poskytuje dostatek informací k tomu, jej útočník řešit vůbec nemusel. Implementaci algoritmu je tedy nutné věnovat maximum pozornosti a stejně tak jeho zabezpečení fyzickému, protože pokud se útočník k modulu nemůže dostat, nemůže ani zkoumat jeho postranní kanály. Proto například šifrovací moduly certifikačních autorit pro podepisování certifikátů jsou zabezpečeny pomocí velmi silné fyzické ochrany.

Bližší informace o postranních kanálech lze hledat například v [1].

# 5 APLIKACE PRO PODPORU VÝUKY

Součástí této práce je aplikace určená pro podporu výuky kryptosystémů založených na problému diskrétního logaritmu. Aplikace má za cíl jednoduchou a srozumitelnou formou představit jednotlivé kryptosystémy využívající problému diskrétního logaritmu a jejich funkce tak, jak jsou popsány v textu výše. V této kapitole bude představena aplikace z provozního hlediska, uživatelské rozhraní a funkce budou rozebrány v kapitole 6.

## 5.1 Aplikační platforma

Základním předpokladem projektu bylo vytvořit jednoduchou, přenositelnou aplikaci přístupnou bez nutnosti instalace či konfigurace na klientském počítači. Aplikace měla být také nezávislou na prostředí, ve kterém je provozována.

Vzhledem k dostatečnému rozšíření a výkonu sdělovacích sítí a oblibě sítě Internet, bylo vybráno webové rozhraní jako přístupový prostředek k aplikaci přes protokol http. Díky tomu lze aplikaci provozovat v libovolném operačním systému s podporou grafického prostředí na webovém prohlížeči splňujícím standardy pro vykreslováním webu, tedy především na prohlížečích Firefox a Opera. Vzhledem k tomu, že Internet Explorer dlouhodobě tyto standardy nedodržuje, není pro něho aplikace optimalizována a v případě jeho použití se mohou vyskytnout chyby při vykreslování uživatelského rozhraní.

Jako implementační nástroj byl vybrán programovací jazyk Java, jehož vlastnostem je věnována samostatná kapitola. Použití tohoto jazyka zajišťuje multiplatformnost jak pro poskytovatele aplikace, tak pro jejího uživatele.

## 5.2 Programové prostředí Java

Java je objektově orientovaný programovací jazyk podobný například C++ nebo C#. Byla navržena tak, aby byla snadno přenositelná na různé počítačové platformy, což je umožněno koncepcí, kdy se zdrojový text kompiluje do strojově nezávislého, ale velmi efektivního bajtového kódu (bytecode). Ten je potom kompilován na cílovém počítači pomocí JVM (Java Virtual Machine) na libovolné počítačové platformě podporující JRE (Java Runtime Environment).

Výhodou jazyka je také jeho otevřená licence GPL, resp. GPLv2, pod kterou jsou šířeny hlavní Java součásti. Za vývojem koncepce jazyka Java stojí společnost Sun Microsystems/Oracle s vývojovým prostředím NetBeans a společnost IBM s prostředím Eclipse. Díky svojí uživatelské přívětivosti, kdy některé obslužné části kódu zajišťuje bez uživatelova vědomí (správa alokace paměti, garbage collector, konstruktory atd.) a podpoře zmíněných softwarovým gigantů se Java stala velmi rozšířenou technologií a na jejím vývoji a podpoře se neustále pracuje.

Do světa webových stránek přinesla Java interaktivitu do té doby veskrze statických stránek a podporu zpracování kódu aplikací na straně klienta, což značným způsobem odlehčuje zátěž serverů a celkovou náročnost komunikace mezi dvojicí klient-server. Vzhledem k tomu, že již při návrhu byl kladen důraz na bezpoečnost aplikací, je ještě před spuštěním kódu ověřena syntaxe, čímž se zamezí pádům programu v důsledku poškozeného nebo neodborně modifikovaného kódu. Program vytvořený v Javě nemá povolen přístup k okolním programům, čímž se vylučuje zavedení Java virů a podobného malware.

Hlavní výhodou, kterou však Java nabízí v porovnání s ostatními programovacími jazyky, je obrovské množství podpůrných knihoven, vytvářených jak hlavními vývojáři v Sun Microsystems a IBM, tak rozsáhlou komunitou uživatelů. Tyto knihovny dávají Javě jako nástroji velkou vývojovou sílu, protože uživatel může použít části kódu vytvořené jiným uživatelem a nebo si vytvořit knihovnu vlastní, která je pak snadno přístupná a přenositelná v kódu. Jedna z na míru vytvořených knihoven, sloužící pro výpočet zbytku po dělení velkých čísel, zde bude podrobněji rozebrána.

### 5.2.1 Modulo velkých čísel

Matematické operátory v jazyce Java obsahují mimo jiné i operátor pro výpočet zbytku po dělení celých čísel, tzv. modulo. Pro výpočty v kryptografické praxi, kde se počítá s čísly o mnoho řádů vyššími než jsou standardní datové typy programovacích jazyků však tento operátor není použitelný. Problémem je zahazování částí čísel z paměti a tedy výpočty nejsou přesné. Pro výpočet zbytku po celočíselném dělení je tedy nutno použít externí funkce.

Pro tuto ukázkovou aplikaci postačí jednoduchá metoda binárního umocňování [25], kterou lze efektivně počítat i ručně. V programovém prostředí pak ve své základní implementaci umožní výpočet v řádech čísel přibližně $3^{2300}$ mod 750, což pro ukázkové účely zdaleka postačuje. Metoda bude vysvětlena na příkladu.

Je-li zadán příklad $7^{18}$ mod 11, je tento příklad ručně neřešitelný. Metoda binárního umocňování postupuje následovně:

Exponent $k = 18$ převede do binárního tvaru $k = (10010)_b$. Nejnižší bit označí $k_0$, vyšší $k_1$, …

S generátorem $b = 7$ pak pracuje tak, že:

$b_0 = 7^1 \bmod 11 = 7;$ $k_0 = 0$

$b_1 = 7^2 \bmod 11 = 5;$ $k_1 = 1$

$b_2 = 5^2 \bmod 11 = 3;$ $k_2 = 0$

$b_3 = 3^2 \bmod 11 = 9;$ $k_3 = 0$

$b_4 = 9^2 \bmod 11 = 4;$ $k_4 = 1$

Z vypočtených parametrů určí výsledek tak, že mezi sebou vynásobí parametry bn, pro které odpovídající parametr kn je nenulový a z výsledku součinu určí zbytek po dělení zadaným číslem. Tedy $c = b_1.b_4 \bmod 11 = 5.4 \bmod 11 = 9$. Ověřením na kalkulačce 1628413597910449 mod 11 = 9.

Metoda je velice efektivní ve výpočtech s nízkým modulem, nejvýše několika stovek. Ve skutečných implementacích šifrovacích algoritmů se využívá Malá Fermatova věta další matematické operace, jejichž popis je nad rámec tohoto textu.

## 5.3 Struktura aplikace

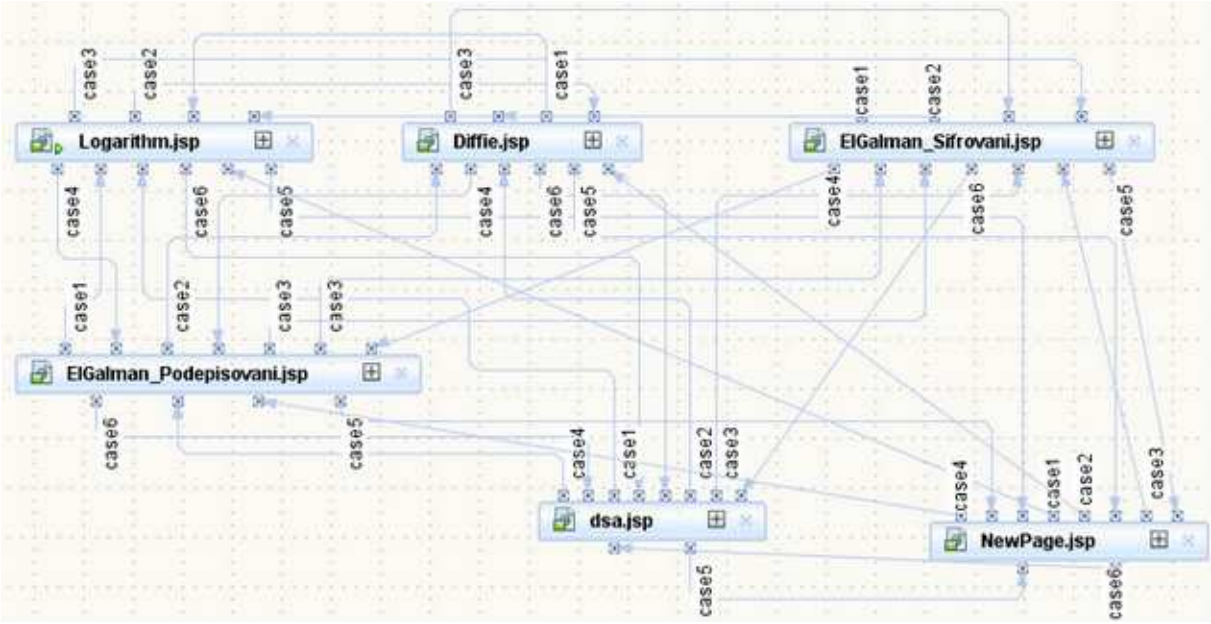
Programovací prostředí NetBeans pro vývoj Java aplikací nabízí při vytváření nového projektu volbu Java web. Přednastavený zdrojový kód tedy přímo vytváří základ budoucí webové aplikace. Další funkce, které pak prostředí Java nabízí umožňuje efektivní návrh grafického uživatelského rozhranní GUI se základními prvky, které se mohou v aplikaci vyskytovat. Je také možné vytvářet prvky vlastní, s vlastním vzhledem a chováním.

Vytvořená aplikace se skládá z několika nezávislých modulů, které jsou vzájemně provázány tak, aby plnily požadovanou funkci. Díky tomu nemusí program obsahovat duplicitní kód a stává se kompaktnější. To je pro aplikaci pracující v síťovém prostředí důležitá vlastnost, protože klesají nároky na přenos dat.

Dílčí aplikační bloky, které jsou na sobě funkčně nezávislé, ale vzájemně na sebe ve výsledné aplikaci odkazují formou hypertextového odkazu, umožňuje Java propojit v jednoduchém grafickém prostředí, jak ukazuje Obr. 7. Výhodou tohoto přístupu je značná

jednoduchost ž´řešení, nevýhodou pak malá přehlednost z případě tvorby rozsáhlých aplikací s velkým množstvím oddělených částí.

Každý tento blok lze na aplikačním serveru provozovat i samostatně bez odkazů na ostatní bloky, což je další výhoda aplikace. Bloky, které časem pozbudou svůj význam lze z aplikace velice snadno odstranit, aniž by funkčnost ostatních bloků byla narušena. Stejným způsobem pak lze přiřadit bloky nové. Jeden náhradní blok je již v aplikaci použit a lze jej snadno využít pro rozšíření aplikace v případě budoucích potřeb.




*Obr. 7: Propojení dílčích částí aplikace*

## 5.4 Aplikační server

Daná aplikace pro podporu výuky kryptosystémů založených na problému diskrétního logaritmu tak jak je vytvořena není spustitelná přímo. Pro její běh je nutné k ní přistupovat přes webový prohlížeč zadáním URL adresy vedoucí na webový server. Tento server nemůže být jednoduchým řešením jaké je příklad integrováno v serverových operačních systémech společnosti Microsoft, ale musí jít o komplexní aplikační server s podporou Java aplikací.

Aplikačních serverů schopných efektivně splnit funkci serveru pro navrženou aplikaci je několik, mezi nejznámější patří bezesporu JBoss, Apache Tomcat a GlassFish. Poslední jmenovaný je proprietálním řešením od společnosti Sun Microsystems, šířeným pod licencí GPLv2 a tedy zcela zdarma. Software je dostupný pro běh na všech systémech MS Windows, Linux a Unix, jde tedy o velmi univerzální nástroj a proto bude představen blíže.

GlassFish server je možné stáhnout přímo ze stránek výrobce pro danou platformu. Instalace je dostatečně intuitivní, nastaví se pouze umístění instalace, administrátorský účet chráněný heslem a výchozí porty pro administraci serveru a port, na kterém bude server naslouchat pro požadavky klientů. Pro administraci je výchozí port 4848, pro aplikace pak port 8080. Tyto porty lze samozřejmě změnit podle konkrétních požadavků. Z administrativní konzole je třeba nastavit publikaci webové aplikace, v tomto případě ./Cryptolearing/dist/Cryptolearning.war a zvolenou URL adresu za doménovým jménem. Po provedení těchto nastavení je aplikace přístupná pro všechny klientské stanice, které se přihlásí na danou adresu.

Výhodou řešení za pomoci aplikačního serveru je přístupnost k aplikaci z jakéhokoli počítače buď v lokální síti, do které patří adresa nainstalovaného serveru a nebo v případě použití serveru s veřejnou IP adresou, přístup odkudkoli z internetu. Přístup lze samozřejmě omezit přímo nastavením na aplikačním serveru, buď pouze zadáním přístupového hesla a nebo kombinace jména a hesla. V základu však pro tuto možnost není aplikace navržena.

Přesný popis instalace aplikačního serveru GlassFish, publikace aplikace a základní nastavení jsou detailně popsána v příloze 1.

# 7 ROZHRANÍ WEBOVÉ APLIKACE

Navržená aplikace poskytuje webové moduly pro představení problematiky diskrétních logaritmů, ElGamalův algoritmus pro šifrování zpráv, ElGamalův algoritmus pro podepisování zpráv, Diffie-Hellmanův algoritmus pro ustavení klíče a jednoduchý modul vysvětlování podstatu podepisování zpráv s pomocí certifikátů. Včechny tyto moduly mají jednotné základní rozhraní, se kterým uživatel pracuje. V této kapitole bude představeno jednak toho společné rozhraní, tak jednotlivá rozhraní dílčí.

## 7.1 Základní nabídka

Základní částí celé webové aplikace je nabídka jednotlivých aplikačních modulů. Tvoří celou horní část aplikace a je neměnná pro všechny moduly. Její jedinou funkcí je navigace mezi dílčími částmi aplikace. Ve své funkční podstatě pracuje jako panel s hypertextovými odkazy.

Pro jeho implementaci je použit vytvořený balíček s názvem Topmenu.jspf. Jakákoli změna v tomto balíčku se automaticky projeví ve všech grafických rozhraních modulů. Vzhled této části aplikace ukazuje Obr. 8.


*Obr. 8: Základní menu aplikace*

## 7.2 Generátor prvočísel

Dalším blokem, který je společný všem částem aplikace je blok generátoru prvočísel. Tento blok slouží jako pomocník pro uživatele, který pro výpočty jednotlivých šifrovacích logaritmů potřebuje volit různá prvočísla. Blok umožňuje generovat prvočísla v rozsahu čísel Integer, pro praktické výpočty v ukázkách však většinou stačí prvočísla v rozsahu 0 – 300.

Programový kód bloku je uložen v balíčku PrimeGen.jspf a využívá externě vytvořené knihovny PrimeGen.java. Práce s blokem je velice snadná, uživatel pouze zadá rozsah hodnot, ze kterých chce znát prvočísla a stiskem tlačítka „Generuj“ získá náhodná prvočísla z bloku. Grafický vzhled bloku je vidět na Obr. 9.

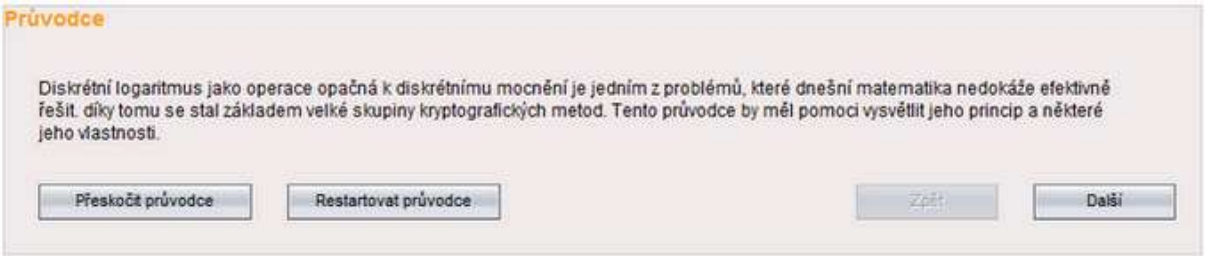

*Obr. 9: Blok generátoru prvočísel*

## 7.3 Průvodce

Blok průvodce je dalším blokem s jednotným rozhraním. Jeho úkolem je představit uživateli jednotlivé šifrovací algoritmy tak, jak jsou počítány při praktickém nasazení na obou stranách komunikujících účastníků. Výpočty jsou samozřejmě představeny s hodnotami čísel a parametrů výrazně nižšími, než jaké jsou použitelné pro praktické využití, aby uživatel mohl výpočty zkoušet sám s využitím kalkulačky a ověřovat si výsledky s průvodcem. Průvodce nezobrazuje pro výpočet použité vztahy, ale spíše na aspekty komunikace jako takové a volby jednotlivých parametrů. Pokud uživatel chce zkoušet výpočty sám, může si v jiném okně zobrazit průvodní text algoritmu, který tyto vztahy mimo jiné obsahuje.

Při spuštění modulu je hlavní část vyřazena z provozu, takže průvodce může libovolně přistupovat k jednotlivým částem. Ovládání je tvořeno tlačítky „Zpět“ a „Další,“ a tlačítky pro ukončení průvodce a jeho restart. Funkce tlačítek pro pohyb je vcelku jasná, zajišťují pohyb v průvodci oběma směry tak, aby se uživatel mohl libovolně vracet ke krokům, které mu unikly nebo je přesně nepochopil. Tlačítko pro ukončení průvodce nastaví aplikační modul do interaktivního režimu, ve kterém může uživatel zkoušet vlastní výpočty a v průběhu tohoto funguje jako restartovací tlačítko konkrétního modulu. Tlačítko restartu průvodce naopak vypne interaktivní mód a umožní opětovný průchod algoritmem s pomocí průvodce.

Ačkoli grafické rozhraní průvodce je shodné ve všech modulech, není implementováno jako samostatně vložený blok z důvodu rozdílného ovládacího kódu. Ten umožňuje snadné změny textu průvodce v jednotlivých krocích přímo editací textu v prostředí NetBeans, případně přidání dalších kroků. Grafické rozhraní průvodce modulu pro ukázku diskrétního logaritmu znárodňuje Obr. 10.




*Obr. 10: Grafické rozhraní průvodce*

## 7.4 Test znalostí

Součástí práce je také několik příkladů pro ověření znalostí. Jejich cílem je, aby se uživatel, pracující s aplikací nad problémem zamyslel a našel pokusil se najít řešení zadaného problému.

Testové otázky jsou zadány v dokumentu, který lze otevřít stiskem odkazu „Nápověda“ na testovém rozhraní. Text obsahuje stručný popis algoritmu, vzorce pro jeho výpočet, zadané úkoly k zamyšlení a ukázku výpočtu kryptografického algoritmu. Příklady jsou koncipovány pro výpočet na papír s využitím jak kalkulačky (ideálně kalkulačka v počítači, která přímo obsahuje funkci modulo) tak interaktivního rozhraní webové aplikace.

Pro kontrolu výsledků obsahuje testové rozhraní pole k ověření výsledků jednotlivých příkladů a testů. Aplikace umožňuje také zobrazení správných výsledků, záměrně však neposkytne uživateli přímo návod k řešení úloh.

Grafické rozhraní testové části aplikace zobrazuje Obr. 11.


*Obr. 11: Testové grafické rozhraní*

## 7.5 Modul „Diskrétní logaritmus“

Část navržené webové aplikace týkající se problému diskrétního logaritmu, bez přímé souvislosti s problematikou kryptografie si klade za cíl představit uživateli matematický problém, který současná věda není schopna efektivně řešit. Skládá se ze dvou částí, části diskrétního mocnění a logaritmu a části cyklických grup čísel.

Uživatel by svými pokusy měl zjistit, že výpočet diskrétního mocnění je velice rychlý, ale opačná operacem diskrétní logaritmus, trvá i s malými čísly více času. Měl by si uvědomit, že pro každé číslo nemusí existovat jeho diskrétní logaritmus a že velkou roli zde hraje hodnota modulu, tedy dělitele, který určuje hodnotu zbytku dělení. Z jeho zkoušek by mělo vyplynout, že v určitých skupinách modulů existuje diskrétní logaritmus čísel menšíc než modul vždy a naopak při jiných skupinách modulů může existovat třeba jen několik čísel, které mohou vzniknout jako zbytek po dělení modulem.

Druhá část modulu je věnována již přímo vlastnostem čísel, ve kterých se diskrétní logaritmus čísla určuje, tedy cyklickým multiplikativním grupám. Uživatel si zde má možnost utvrdit poznatky získané v první části aplikace o chování grup v závislosti na volbě modulu a základu (řádu a generátoru). Generováním číselných grup si ověří, že $b^{n-1}$ mod $n$ = 1 pro všechny hodnoty modulu i generátoru a pokud hodnota *n-1* bude dělitelná beze zbytku hodnotou *b*, pak bude v grupě pouze několik hodnot, pro které je definován diskrétní logaritmus, který bude stejný pro různé hodnoty exponentu. Z grafu, který mu aplikace vykreslí, by si měl uvědomit pseudonáhodnost diskrétního logaritmu, tedy že není možné odhadnout, ve které části číselného spektra se hodnota budu nacházet, v případě volby vhodných parametrů.

Pro ověření získaných znalostí má uživatel k dispozici test z otázek v doprovodném textu, který tvoří přílohu č. 2 této práce.

Z vývojového hlediska se tělo webové stránky nachází v balíčku Logarithm.jsp spolu s kódem průvodce a znalostního testu. Aplikace provádí hlavní část výpočtů s využitím knihovny pro výpočet diskrétního mocnění, která je založena na metodě „binárního mocnění“ popsaného v kapitole 5.2.1. Vzhledem k relativní přívětivosti jazyka Java je možné se v kódu snadno orientovat a podle potřeby jej modifikovat.

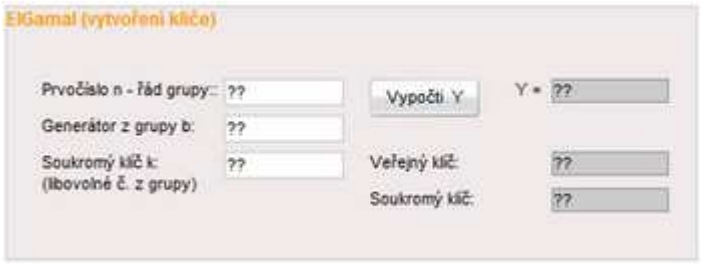
## 7.6 Modul „ElGamalův kryptosystém – šifrování“

Prvním z modulů ElGamalova kryptosystému je sytém pro šifrování zpráv. Jeho cílem je představit metodu založenou na obtížnosti řešení diskrétního logaritmu tak, jak byla navržena panem ElGamalem. Modul zdaleka neslouží pro praktické použití, ale umožňuje uživateli pochopení problému jako celku, při výpočtech s malými čísly.

Modul je opět po spuštění nastaven pro práci s průvodcem, takže si uživatel hned při jeho průchodu osvojí základní poznatky o kryptosystému jako takovém a o práci s grafickým rozhraním. Po ukončení průvodce získá uživatel přístup k interaktivnímu prostředí modulu.

První část je věnována vytvoření klíče ElGamalova kryptosystému. Zde uživatel uplatní část znalostí z modulu věnovaného diskrétnímu logaritmu a zvolí vhodný řád grupy a generátor pro výpočet klíče. Klíč může vypočítat v libovolné grupě, pro šifrování textových zpráv je však nastaven minimální řád grupy na hodnotu 257. To z toho důvodu, že zpráva je tvořena znaky, jejichž číselná hodnota je vyjádřena číslem podle ASCII tabulky. Zpráva je

zde rozdělena a šifrována po znacích, takže řád grupy musí být větší než hodnota jakéhokoli znaku. Grafická podoba oddílu výpočtu klíče je na Obr. 12.




*Obr. 12: Výpočet klíče ElGamalova kryptosystému*

Druhá část je již věnována samotnému procesu šifrování zpráv, přenosu a dešifrování zpráv soukromým klíčem příjemce. Matematický aparát výpočtu je obsažen v doprovodném textu modulu, který tvoří přílohu č.3 této práce. Samotný výpočet není nikterak náročný na postup, jde pouze o aplikaci matematických vzorců, takže jej uživatel snadno pochopí a dokáže využít k řešení úloh, které jsou pro něj připraveny. Svým testováním si ověří, jak se mění hodnoty šifrovaných zpráv a uvidí, jaká slabina vzniká v systému pokud by komunikující strana neměnila náhodný parametr v procesu šifrování.

Z grafického rozhraní, zobrazeného na Obr. 13 také jasně uvidí, které parametry musí mít obě strany, aby byla šifrovaná komunikace možná a bezpečná. V aplikaci není žádným způsobem řešena otázka autorizace zprávy, jde o ukázku zajištění důvěrnosti zprávy. Autorizaci zpráv řeší ElGamalův systém pro podepisování zpráv, který tvoří další modul aplikace.


*Obr. 13: Rozhraní ElGamalova systému pro šifrování zpráv*

## 7.7 Modul „ElGamalův kryptosystém – podepisování“

ElGamalův kryptosystém pro šifrování zpráv žádným způsobem neřeší autorizaci zpráv, takže příjemce zprávy nemá možnost ověřit, kdo je skutečným autorem zprávy. K tomu slouží podepisování zpráv. Tento modul si klade za cíl představit uživateli mechanizmus podepisování zpráv ElGamalovým kryptosystémem, opět však nezabíhá do úplných detailů vzájemné autentizace uživatelů.

Grafické rozhraní obsahuje stejné sdílené prvky jako ostatní moduly a interaktivní režim je při spuštění také vyřazen. Po průchodu průvodce je uživateli tento režim zpřístupněn a lze začít s testováním. Modul pro výpočet klíče je shodný s výpočtem klíče v modulu šifrování, navíc je zde v průvodci naznačena nutnost nechat si svůj klíč podepsat sertifikační autoritou a získat tak certifikát. Novým modulem je zde však část pro výpočet hashe zpráv.


*Obr. 14: Rozhraní Shamirova hashovacího algoritmu*

Obr. 14 znázorňuje rozhraní pro výpočet hashe zprávy pomocí Shamirova hashovacího algoritmu založeného na problému výpočtu diskrétního logaritmu. Zde uvedená implementace je velice zjednodušená. Pro praktické využití je nutný poněkud odlišný přístup k problému, zde slouží pouze jako ukázka toho, že hashovací funkce nemusí být založena pouze na principu proudového zpracování dat, ale i na čistě početní metodě.

Rozhraní pro výpočet podpisu je také velmi jednoduché stejně jako metoda sama. Uživatel potřebuje pouze svůj soukromý klíč, hodnotu generátoru a řádu cyklické grupy, ze které si soukromý klíč zvolil a nakonec hash přenášené zprávy. Vzorce pro výpočet podpisu jsou opět uvedeny v pracovním textu, který si uživatel z aplikace otevře. Tvoří přílohu číslo č.4. Na straně příjemce pak uživatel musí znovu zadat hash přijaté zprávy, což naznačuje, že příjemce podpisu si jej musí znovu vypočíst ze zprávy, aby mohl podpis ověřit. Skutečnost, že podpis se přenáší v šifrované podobě, aby byla zaručena důvěrnost dat model neřeší, zaměřuje se na princip podpisu.

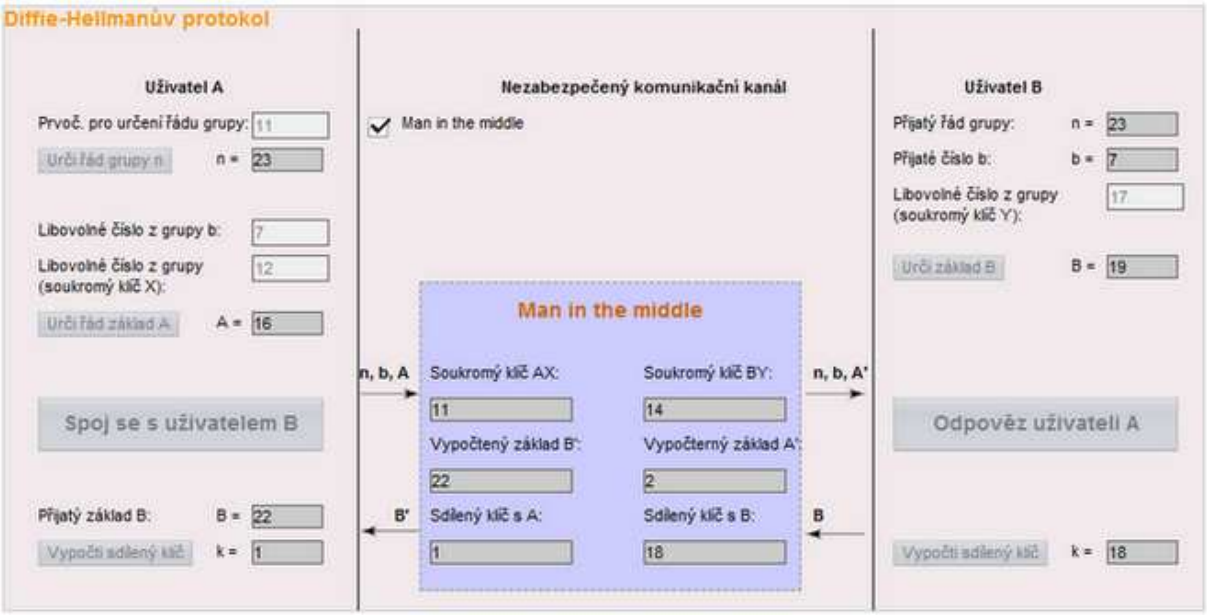
Programový kód bloku je uložen v balíčku ElGamal_podepisovani.jsp, který stejně jako ostatní bloky může být umístěn na aplikační server i samostatně bez ostatních. Pouze odkazy v horní části webového okna by se tak staly nefunkční.

## 7.8 Modul „Diffie-Hellmanův protokol“

Dalším modulem, který aplikace obsahuje je ukázka funkce Diffie-Hellmanova algoritmu pro ustavení klíče mezi dvěma uživateli po nezabezpečených síťových linkách. Aplikace ukazuje uživateli postup, jak celá komunikace probíhá a znázorňuje také největší slabinu Diffie-Hellmanova protokolu, tedy zranitelnost útokem „Man In The Middle.“

Průvodce po spuštění modulu stejně jako v ostatních modulech uživatele provede celým procesem komunikace a seznámí ho s grafickým rozhraním. Doprovodné texty k tomuto modulu opět poskytují trochu teorie o daném protokolu, matematický aparát využívaný protokolem, otázky k zamyšlení a vypracování a ukázku řešené komunikace. Tyto texty tvoří přílohu č. 5.

Grafické rozhraní se znázorněním útočníka „uprostřed“ je na Obr. 15. Je z něho vidět, že uživatel při navazování komunikace neví, že komunikuje s útočníkem a nemůže komunikaci nijak ovlivnit.




*Obr. 15: Grafické rozhraní modulu Diffie-Hellmanova protokolu*

Uživatel při práci s rozhraním opět využije znalostí získaných o cyklických grupách a v příkladech vyzkouší ustavení klíče i jeho zjištění z části zachycených parametrů. Pokud již

uživatel absolvoval ostatní moduly, měl by být také schopen navrhnout ochranu před nastíněným útokem s využitím parametrů podepsaných certifikační autoritou. Ze svojí podstaty by pak ovšem algoritmus byl velmi statický a útočník by jej časem mohl překonat.

## 7.9 Modul „DSA a podepisování zpráv“

Posledním modulem webové aplikace je modul DSA a podepisování zpráv. Tento modul neslouží přímo jako výuková pomůcka, má pouze uživateli přiblížit jedinou oblast, kde se v současné praxi využívá kryptosystémů založených na problému diskrétního logaritmu.

Digital Signature Algorithm (DSA) je systém vycházející ze Shamirova kryptosystému pro podepisování zpráv, odstraňuje však jeho slabiny a definuje delší šifrovací klíče. Modul nepředstavuje přímo metodu výpočtu podpisů touto metodou, protože k ručním výpočtům je metoda složitá a uživatel by si ji stejně nemohl prakticky ověřit. Na místo toho je popsán postup získání certifikátu a jeho použití pro autorizaci zpráv. Je popsán v praxi nejčastější model, kdy pouze jeden z účastníků má vlastní certifikát a ověření identit je tedy pouze jednostranné. Ve většině případů je situace taková, že se autorizuje server směrem k uživateli, který si tak ověří, zda komunikuje opravdu se serverem, se kterým zamýšlel (ochrana například proti „pharmingu“).

Systém DSA a metoda, jakou počítá svoji dvojici klíčů je popsána v doprovodném textu, který lze z modulu otevřít. Je zde uveden i postup podepsání a ověření zprávy, včetně jednoduchého příkladu výpočtu, cílem však není naučit uživatele vypočítat podpis, ale představit důležitost podepisování zpráv jako takovou.

Programový kód modulu je uložen v balíčku dsa.jsp, odkud je možné snadno upravovat grafickou strukturu rozhraní, komentáře průvodce či testové otázky.

# 8 ZÁVĚR

Tento projekt v krátkosti představuje vývoj kryptografie od jejích, z dnešního pohledu, primitivních počátků až po současnost, kdy skupiny symetrických a asymetrických kryptosystémů poskytují komplexní řešení bezpečnosti pro přenos zpráv.

Práce se zaměřuje na asymetrickou kryptografii, jejíž bezpečnost je založena na neřešitelnosti některých matematických problémů. Představen je například problém faktorizace, který je využit v nejrozšířenějším kryptosystémech, hlavní pozornost je však věnována problému diskrétního logaritmu.

Problematika diskrétního logaritmu zde obsahuje rozbor samotného matematického problému, ale přibližuje také skupiny čísel, ve kterých se diskrétní logaritmus čísla určuje, tedy cyklické multiplikativní grupy. Je zde popsána závislost vlastností diskrétního logaritmu na vlastnostech cyklických grup a parametry grup, v nichž počítané diskrétní logaritmy jsou považovány za NP-úplný matematický problém.

Práce popisuje také metody, které byly navrženy jako řešení diskrétního logaritmu, žádná z nich však nenalézá řešení problému v polynomiálním čase. V krátkosti je představena problematika postranních kanálů, které citelným způsobem narušují bezpečnost kryptografických modulů tím, že neútočí na matematický problém, ale jeho fyzickou implementaci.

Podrobně jsou rozebrány asymetrické kryptografické systémy, které z problematiky diskrétního logaritmu vycházejí. Pozornost je věnována jejich popisu, principu výpočtu zabezpečení a postupu komunikace uživatelů, systémů, který daný algoritmus využívají. Diskutováno je také jejich stávající využití v kryptografické praxi, kdy hlavním oborem nasazení těchto systemuje podepisování zpráv. Představení systémů pro šifrování zpráv, podepisování zpráv, hashování zpráv a ustavení tajného klíče tvoří podstatu navržené aplikace pro podporu výuky.

Tato aplikace se v jednotlivých částech pokouší uživateli vysvětlit a názorně ukázat akce, které jsou v danou chvíli při komunikaci prováděny. Díky interaktivnímu prostředí a řadě praktických úkolů, které jsou uživateli zadány k řešení, lze danou problematiku snadno pochopit a osvojit si její podstatu. Aplikace uživatele vybízí k uvažování nad problémem a k práci s využitím ručních výpočtů. Doprovodné texty s popisem dílčích kryptosystémů, se kterými uživatel pracuje, lze z díky univerzálnímu formátu .pdf z práce snadno vytisknout a použít jako samostatnou studijní pomůcku.

Aplikace jako taková je implementována v jazyce Java, je tedy platformově nezávislá a lze ji použít na jakémkoli grafickém operačním prostředí. Umožňuje také přenést nutný výpočetní výkon na stranu klientské stanice, což odlehčuje zátěž serveru. Aplikace pro svoji funkci využívá libovolného aplikačního serveru (GlassFish, Apache Tomcat,…) a uživatelé k ní přistupují přes webový prohlížeč. Díky svému modulovému zpracování lze jednotlivé části aplikace snadno modifikovat a obměňovat tak například zadané otázky a jejich vyhodnocení nebo třeba ukázky výpočtu a řešených příkladů.

# POUŽITÁ LITERATURA


[1] KŘÍŽ, Jiří <xkrizj01@stud.feec.vutbr.cz>. Postranní kanály v kryptografii. Květen 2007. [bakalářská práce, citace str.2].

[2] KLÍMA, Vlastimil <v.klima@volny.cz>. Hašovací funkce, principy, příklady a kolize. 19. 3. 2005. [.pdf dokument]. Dostupný z: http://www.cryptofest.cz/slides/klima_cryptofest_2005.pdf.

[3] BURGET, Radim < burgetrm@feec.vutbr.cz >. Vyčíslitelnost a složitost P, NP a NPC. [.pdf přednáška předmětu MTIN]. Dostupný z: http://adela.utko.feec.vutbr.cz/index.php?option=com_content&task=view&id=17&Itemid=27.

[4] HOUSER, Pavel. Faktorizace na DNA počítači. 22. 4. 2002. [online]. Dostupný z: http://www.root.cz/clanky/faktorizace-na-dna-pocitaci/.

[5] ROBLES, Silvia. The RSA Cryptosystem. 9. 5. 2006. [.pdf dokument]. Dostupný z: http://ocw.mit.edu/NR/rdonlyres/Mathematics/18-304Spring-2006/A9ACFD84-044C-4903-A5FB-156F42B8C0EF/0/rsa_robles.pdf.

[6] DAVIS, Tom. RSA Encryption. 10. 10. 2003. [.pdf dokument]. Dostupný z: http://www.geometer.org/mathcircles/RSA.pdf.

[7] CARTER, A. Brian. Asymetric Cryptosystems. 27. 8. 2007. [.pdf dokument]. Dostupný z: http://briancarter.info/pubs/asymmetric_cryptosystems.pdf.

[8] ODLYZKO, A. M. Discrete Logarithms In Finite Fields And Their Cryptographic Significance. AT&T Bell Laboratories. [.pdf dokument]. Dostupný z: http://www.dtc.umn.edu/~odlyzko/doc/arch/discrete.logs.pdf.

[9] BRUGGER, F. Max. The Discrete Logarithm Problem and Ternary Functional Graphs. 9.8.2007. [.pdf dokument]. Dostupný z: http://www.rose-hulman.edu/mathjournal/archives/2007/vol8-n2/paper8/v8n2-8pd.pdf.

[10] WEISSTEIN, Eric. Group. [online]. Dostupné z: http://mathworld.wolfram.com/Group.html.

[11] METCALF, Luke. ElGamal Discrete Log Cryptosystem. [online encyklopedie]. Dostupný z: http://www.nationmaster.com/encyclopedia/ElGamal-discrete-log-cryptosystem.

[12] PINKAVA, Jaroslav <jaroslav.pinkava@aec.cz>. Nové směry v kryptografii s veřejným klíčem. [.pdf dokument]. Dostupný z: http://crypto-world.info/pinkava/konference/virus.pdf.

[13] SUTHERLAND, V. Andrew. Order Computations in Generic Groups. Červen 2007. [.pdf dokument]. Dostupný z: http://groups.csail.mit.edu/cis/theses/sutherland-phd.pdf.

[14] CORON, Jean Sébastien. A New Baby-Step Giant-Step Algorithm and Some Applications to Cryptanalysis. [.pdf dokument]. Dostupný z: http://www.ssi.gouv.fr/fr/sciences/fichiers/lcr/colepo05.pdf.

[15] STUDHOLME, Chris. The Discrete Log Problem. 21. 7. 2002. [.pdf dokument]. Dostupný z: http://www.cs.toronto.edu/~cvs/dlog/research_paper.pdf.

[16] GUAN, D J. Pollard's Algorithm for Discrete Logarithm Problem. 25. 8. 2003. [.pdf dokument]. Dostupný z: http://icpc.baylor.edu/past/icpc2004/RegReport/guan.cse.nsysu.edu.tw/data/pollard.pdf

[17] MENEZES, J. Alfred. OORSCHOT C. Paul. VANSTONE A. Scott. Handbook of applied cryptography. Září 1996. ISBN: 0-8493-8523-7. CRC Press.

[18] MENEZES, Alfred. USTAOGLU, Berkant. On Reusing Ehemeral Keys In Diffie-Hellman Key Agreement Protocols. [.pdf dokument]. Dostupný z: http://www.math.uwaterloo.ca/~ajmeneze/publications/ephemeral.pdf.

[19] ROSA, Tomáš < trosa@ebanka.cz >. KLÍMA, Vlastimil < klima@lec.cz >. Kryptologie pro praxi. Červen 2004. [.pdf dokument]. Dostupný z: http://crypto-world.info/klima/2004/st_2004_06_12_12.pdf.

[20] ELGAMAL, Tather. A Public Key Cryptosystem And A Signature Scheme Based On Discrete Logarithms. 1998. [.pdf dokument]. Dostupný z: http://groups.csail.mit.edu/cis/crypto/classes/6.857/papers/elgamal.pdf.

[21] SINDEREK, Ralf. A Discrete Logarithm Hash Function for RSA Signatures. [.pdf dokument]. Dostupný z: http://senderek.com/SDLH/discrete-logarithm-hash-for-RSA-signatures.ps.

[22] ACAN, Huseyin, KAYA, Kamer, SELCUK, Ali. Capture Resilient ElGamal Signature Protocols. [s.l.]: Springer Berlin / Heidelberg, c2006. 951 s. ISBN 978-3-540-47242-1.

[23] KAMMER, G. Raymond. Digital Signature Standard. U.S. Department Of Commerce. 27. 1. 2000. [.pdf dokument]. Dostupný z: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.

[24] SCHNEIER, Bruce. Applied cryptography : Protocols, Algorithms and Source Code in C. 2nd edition. [s.l.] : John Wiley & Sons, Inc., c1996. 758 s. ISBN 0-471-11709-9.

[25] PŘIKRYL , Jan, VLČEK, Miroslav. Aplikace modulární aritmetiky : Algoritmus RSA. [online]. 2007. Dostupný z WWW: <http://euler.fd.cvut.cz/predmety/ma/files/ma-06-2007.pdf>.

# Seznam použitých zkratek

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| D-H protokol | Diffie - Hellmanův protokol |
| DL | Diskrétní Logaritmus |
| DSA | Digital Signature Algorithm |
| DSS | Digital Signature Standard |
| GPL | General Public License |
| GPLv2 | General Public License verze 2 |
| GUI | Graphical User Interface |
| IBM | International Business Machines |
| JRE | Java Runtime Environment |
| JVM | Java Virtual Machine |
| MD5 | Message Digest 5 Algorithm |
| MS | Microsoft |
| NP problém | Problém s řešením v Nederministickém Polynominálním čase |
| P problém | Problém s řešením v Polynominálním čase |
| RSA | Asymetrický kryptosystém Rivest - Shamir - Adlemann |
| SHA1, SHA2 | Secure Hash Algorithm |
| URL | Uniform Resource Locator |
| XOR | Exclusive Or |

# Seznam příloh

Příloha 1: Instalace a nastavení aplikačního serveru

Příloha 2: Texty k modulu diskrétního logaritmu

Příloha 3: Texty k modulu ElGamal - šifrování

Příloha 4: Texty k modulu ElGamal - podepisování

Příloha 5: Texty k modulu Diffie-Hellmanův protokol

Příloha 6: Texty k modulu Diffie-Hellmanův protokol

Příloha 7: Disk CD s elektronickou verzí práce a navrženou web aplikací

# Příloha 1: Instalace a nastavení aplikačního serveru

Webová aplikace pro podporu výuky potřebuje pro svoji funkci nainstalován aplikační server, ze kterého se aplikace spouští na klientských webových prohlížečích. Konkrétní typ aplikačního serveru není nutný, lze použít libovolný software. Zde bude popsána instalace software GlassFish v2.1. Jde o freeware aplikační server vyvinutý společností Sun Microsystems, šířený pod licencí GPLv2. Software lze stáhnout přímo ze stránek výrobce a jeho instalační soubor pro platformu Windows je umístěn i v příloze 7.

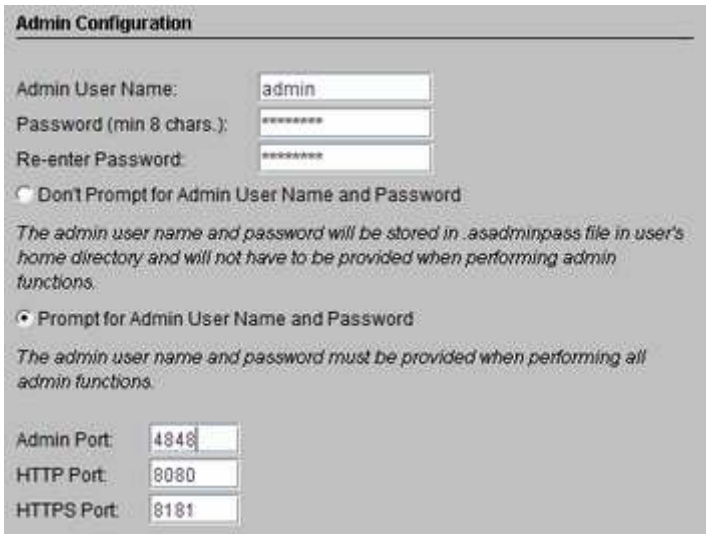
Instalační soubor pod platformou Windows lze spustit přímo dvojklikem. Pro správný chod je třeba mít v počítači nainstalován JRE. Vzhledem k tomu, že dnes jde již o standardní vybavení počítače, lze předpokládat, že je nainstalováno.

Spustí se klasický instalační průvodce od společnosti Sun Microsystems. Tlačítkem další se přejde k licenčnímu ujednání, které je vhodné prostudovat a přijmout. V dalším kroku je uživatel dotázán na cílové umístění pro instalaci software. Umístění záleží pouze na uživateli


Dalším dotazem průvodce je umístění Java 2 SDK software, který je pro běh serveru nezbytný. Pokud není nainstalováno, je třeba jej nainstalovat také. Instalační soubor pro Windows je součástí přílohy č. 7.


Po nastavení cesty a stisku tlačítka další je uživateli zobrazeno rozhraní pro nastavení základních vlastností serveru. Nastavuje se účet administrátora serveru, tradičně „admin“ a jeho přístupové heslo. To je nutné si zapamatovat. Volba, zda se software bude na heslo při přístupu z webového prohlížeče serveru dotazovat, záleží pouze na uživateli.

Je nutné také zvolit porty, na kterých bude server poslouchat pro příchozí požadavky. Nastavuje se port pro přístup k administrační konzoly a porty pro http a https. Volba záleží opět na uživateli, je třeba nastavit porty, na kterých nenaslouchá jiná aplikace.




Další obrazovka slouží k některým nastavením serveru, jako jeho aktualizace apod. Vhodné je povolit aktualizace, přidat složku bin do proměnné *path* a vytvořit služby Windows. Instalace pak již pouze zkontroluje, zda je dostatek místa na disku a instalátor spustí samotnou instalaci.

Po jejím proběhnutí ještě instalátor nabídne nepovinnou instalaci a následně již lze server spustit tlačítkem „Start Server.“ Spuštění se ohlásí upozorněním uživateli.

Nyní je již možné přistoupit ke konfiguraci serveru. Přístup je přes webový prohlížeč, v případě že se nastavuje lokálně, přistupuje se přes URL: *localhost:4848*. Aplikační server pak v závislosti na zvoleném nastavení zobrazí výzvu k zadání administrátorského hesla.


Otevře se rozhraní pro administraci serveru. Pro nastavení vytvořené aplikace je třeba jít do záložky „Web application“ v levém menu obrazovky.


Zde po stisku tlačítka „Deploy“ se objeví rozhraní pro zveřejnění aplikace serverem. V tomto případě stačí ponechat základní nastavení pro webové aplikace (.war) a nastavit cestu k souboru webové aplikace, který se nachází ve složce aplikace a její podložce *dist*.

(.\Cryptolearning\dist\Cryptolearning.war). Dále se nastaví jméno aplikace tak, jak se k ní bude přistupovat z webového rozhraní. Pro jistotu volte název s pouze malými písmeny.


Potvrzením tlačítkem OK se aplikace zveřejní a je přístupná přes webový prohlížeč jak lokálně, tak ze sítě na adrese počítače. Pokud byl pro server určen jiný port než 80, je třeba v adresním řádku tento port specifikovat.

# Příloha 2: Texty k modulu diskrétního logaritmu

**Teorie:**

Diskrétním logaritmus je označení pro jeden z několika NP-úplných problémů používaných v kryptografické praxi pro tvorbu kryptosystémů. Jde o problém, který by se dal popsat jako problém nalezení takového čísla *k*, pro které platí

$$b^k \bmod n = g,$$

tedy řešení rovnice

$$log_b g = k \ (mod\ n)$$

Řešení se na první pohled zdá jednoduché, avšak matematická věda zatím nezná žádný efektivní algoritmus, jak daný problém vyřešit v nepolynomiálním čase. Pro lepší pochopení poslouží následující příklad.

Jsou zadány čísla $b = 4$, $k = 6$ a $n = 11$. S těmito čísly zabere vyřešení první rovnice jen pár vteřin, jde o prosté umocnění a výpočet zbytku po dělení celým číslem,
$g = b^k \bmod n = 4^6 \bmod 11 = 4$.

Nyní se situace změní a zadány jsou čísla $b = 4$, $g = 4$ a $n = 11$. Dosazením do druhé rovnice dostaneme $\log_4 4 = k \ (mod\ 11)$, což je číselně $1 = k \ (mod\ 11)$. Zjevná chyba je způsobena tím, že číslo *g* nevzniklo přímo mocněním, ale jde o výsledek celočíselného dělení čísla, vzniklého mocněním, číslem 11. Obecně existuje nekonečně mnoho různých čísel ($b^k$), jejichž zbytek po celočíselném dělení číslem 11 je 4.

Výsledky operace výpočtu zbytku po celočíselném dělení, modulo, se opakují s násobky intervalu <0, *n*>. Tato vlastnost definuje tzv. Cyklické grupy. Diskrétní logaritmus se tedy vždy počítá v dané cyklické grupě a protože hodnota základu pro výpočet zbytku po dělení vzniká umocněním čísla, mluvíme o Cyklické multiplikativní grupě. Vlastnosti těchto grup pak zásadním způsobem určují vlastnosti celého diskrétního logaritmu.

Hodnoty čísel v multiplikativní cyklické grupě jsou vždy v intervalu <0, *n*> kde n se nazývá řádem grupy. Hodnoty po sobě jdoucích čísel v grupě jsou náhodné, ne všechny hodnoty z daného intervalu se grupě musí vyskytovat. Z toho že ne všechny hodnoty musí v grupě existovat plyne, že diskrétní logaritmus nelze určit pro jakákoliv čísla, pouze je-li základem grupy prvočíslo, pak existuje diskrétní logaritmus pro všechna čísla zmíněného intervalu.

**K procvičení:**

a) Ručně, resp. s využitím kalkulačky, zkuste určit mocninu čísla $13^{48}$ mod 53 [výsledek: 44]

b) Ručně, resp. s využitím kalkulačky, zkuste určit všechny mocnitele *k* v grupě řádu *n* = 12 tak, aby hodnota mocniny byla 7. Jako generátor (základ) využijte rovněž číslo 7.

c) Využitím vykreslení grafu v aplikaci zkuste změnou hodnoty generátoru a řádu grupy odvodit, jaké vlastnosti by měli tyto hodnoty mít, aby byly dobře použitelné v kryptografii.

d) Vypočtěte všechny prvky grupy, kde *b* je celočíselným dělitelem (*n* -1). (b=12, n=37). V čem je taková grupa zvláštní? Hodí se pro využití v kryptografii?

[Tip: Kalkulačka ve Windows obsahuje funkci přímo pro výpočet modulo]

**Otázky a úkoly:**

1) Kolik metod dokáže řešit problém diskrétního logaritmu v lineárně rostoucím čase v závislosti na zvětšujících se parametrech cyklických grup?

A) 0 B) 1 C) 2

2) Vlastnosti diskrétního logaritmu jsou dány především:

A) Velikostí exponentu, jehož hodnota se hledá

B) Vlastnostmi cyklických grup

3) S využitím aplikace nalezněte takovou hodnotu mocniny, pro kterou v grupě s řádem n = 9 a generátorem b = 2 neexistuje diskrétní logaritmus.

4) Kolik různých hodnot může mocnina z předchozího zadání nabývat, aby pr ni byl definován diskrétní logaritmus?

5) Hodnoty mocnin při výpočtu cyklických grup po sobě následují:

A) V předem známém pořadí

B) Náhodně

C) Pseudonáhodně

6) Jakých hodnot by měli nabývat parametry n a b, aby cyklická grupa byla vhodná pro použití v kryptografii?

A) *n* je prvočíslo, *b* je prvočíslo

B) *n* není prvočíslo, *b* je prvočíslo

C) *n* je prvočíslo, vlastnosti *b* určuje konkrétní kryptografické schéma

# Příloha 3: Texty k modulu ElGamal - šifrování

**Teorie:**

Jde o asymetrický kryptosystém založený na problému diskrétního logaritmu. Lze jej využít buď k šifrování nebo k podepisování zpráv. Dnes již z bezpečnostního hlediska původní návrhy nelze použít, přesto některé protokoly nebo aplikace nabízejí volbu upravených ElGamalových algoritmů. K masivnímu rozšíření nedošlo také proto, že šifrovaná nebo podepsaná zpráva algoritmem ElGamal má dvojnásobnou délku oproti zprávě původní.

**Vytvoření dvojice klíčů:**

4) Vygeneruje náhodné velké prvočíslo *n* a generátor *b* multiplikativní grupy $(\mathbf{Z}_n)^x$ celých čísel modulo *n*.
5) Zvolí náhodné celé číslo k, pro které platí $1 \le k \le n-2$ a vypočte $b^k \bmod n$.
6) Veřejný klíč je pak ($n$, $b$, $b^k$), soukromý klíč potom $k$.

**Šifrovaná komunikace probíhá následovně:**

6) Komunikující strana si obstará veřejný klíč protistrany z důvěryhodného zdroje
7) Rozdělí zprávu do více úseků, označených M, pro které platí, že číselná velikost M je v rozsahu $1 \le M \le n-1$
8) Zvolí si náhodné číslo $x$ z intervalu $1 \le x \le n-2$ tak, aby bylo nesoudělné s *n-1*
9) Vypočte $\gamma = b^x \bmod n$ a $\delta = M.(b^k)^x \bmod n$
10) Odešle šifrovaný text $c = (\gamma, \delta)$

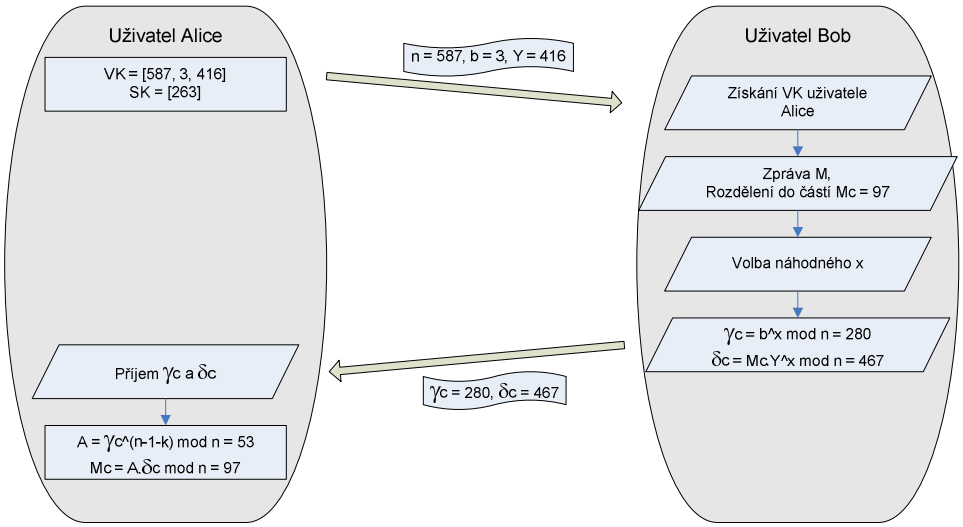
**Příjemce (vlastník privátního klíče) přijme šifrovaný text *c*:**

3) Použije privátní klíč $k$ pro výpočet $A = \gamma^{n-1-k} \bmod n$ (poznámka: $\gamma^{n-1-k} = \gamma^{-k} = b^{-kx}$)
4) Vypočte původní hodnotu $M$ jako $M = (\gamma^{-k}).\delta \bmod n$.

Výhoda ElGamalova kryptosystému pro šifrování oproti například RSA je v tom, že do šifrovacího procesu vnáší jistou náhodnost volbou parametru $x$. Na druhou stranu je však nutné tento parametr v průběhu komunikace neustále náhodně měnit a po použití důkladně zničit.

Praktická komunikace pak probíhá tak, jak je naznačeno na obrázku. Je vidět, že slabinou v tomto jednoduchém schématu je přenos veřejného klíče mezi uživateli. Pokud by někdo komunikaci zachytil a veřejný klíč vyměnil za svůj vlastní, byl by jediným, kdo zná

soukromí klíč a mohl by zprávu snadno dešifrovat. Z tohoto důvodu se k distribuci klíčů využívá certifikátů podepsaných certifikační autoritou.




**K procvičení:**

a) S použitím vzorců uvedených výše, zkuste zašifrovat a dešifrovat krátkou zprávu s použitím kalkulačky. Pro jednodušší výpočty volte malé hodnoty jednotlivých parametrů.

[Tip: Kalkulačka ve Windows obsahuje funkci přímo pro výpočet modulo]

**Otázky a úkoly:**

1) Kolik parametrů tvoří veřejný klíč uživatele v kryptosystému ElGamal?

2) Pokud by se neměnila hodnota náhodného parametru x, měnila by se při šifrování hodnota parametru Gama v závislosti na změně zpráv?

A) Ano
B) Ne
C) Záleží na velikosti změny zprávy

3) Pokud při přenosu zprávy dojde k bitové chybě a je změněna hodnota některé z přenášených zpráv, je možné za určitých podmínek zprávu dešifrovat? (neuvažujte opravné mechanizmy přenosu zpráv)

A) Ano
B) Ne
C) Pouze pokud došlo k záměně bitu LSB řádu grupy v přenášeném klíči

4) Vypočtěte hodnotu parametru Delta, je-li veřejný klíč uživatele {257, 31, 199}, šifrovaná zpráva je znak „s“ (ASCII hodnota 115) a parametr $x = 7$.

5) Pokud váš soukromý klíč je $k$ = 127 a parametry Delta = 130, Gama = 248, jaká zpráva byla přenesena? Do výsledkové tabulky zaneste číselnou hodnotu, ne ASCII.

**Příklad výpočtu:**
Zadání: Zašifrujte zprávu M = 208 ElGamalovým kryptosystémem. Řád multiplikativní grupy zvolte $n$ = 257, ostatní parametry volte libovolně. Zprávu poté i dešifrujte.

*Tvorba klíče*:

- volba $b$ = 38, $k$ = 174. Pak:

$$b^k \bmod n = 38^{174} \bmod 257 = 178$$

- klíče: SK = {174}, VK = {257, 38, 178}

*Šifrování*:

- zpráva M = 208, obdržený VK = {257, 38, 178}
- volba x = 4 …. Neodpovídá podmínce o nesoudělnosti s $n$ – 1
- volba nového x = 40.

$$\gamma = b^x \bmod n = 38^{40} \bmod 257 = 136$$

a

$$\delta = M.y^x \bmod n = 208.178^{40} \bmod 257 = 26$$

- odeslaný kryptogram C = {136, 26}

*Dešifrování*:

- přijatý kryptogram C = {136, 26}, známý SK = {174}

$$A = \gamma^{n-1-k} \bmod n = 136^{257-1-174} \bmod 257 = 8$$

a

$$M = A.\delta \bmod n = 8.26 \bmod 257 = 208$$

- přijatá zpráva M´=208

# Příloha 4: Texty k modulu ElGamal - podepisování

**Teorie:**

ElGamalův systém pro podepisování zpráv je jedním z prvních systémů pro podepisování vůbec. Jeho bezpečnost je opět založena na problému diskrétního logaritmu, ale pro současné bezpečnostní potřeby již nedostačuje. Jde o výpočetně poměrně nenáročný systém, proto zde je podrobně představen.

**Vytvoření dvojice klíčů:**

4) Vygeneruje náhodné velké prvočíslo *n* a generátor *b* multiplikativní grupy $(\mathbf{Z}_n)^x$ celých čísel modulo *n*.
5) Zvolí náhodné celé číslo k, pro které platí $1 \le k \le n-2$ a vypočte $y = b^k \bmod n$.
6) Veřejný klíč je pak (*n*, *b*, *y*), soukromý klíč potom *k*.

**Vytvoření podpisu zprávy:**

5) Zvolí si náhodné, tajné, celé číslo x, pro které platí $1 \le x \le n-2$ a je nesoudělné beze zbytku s *n-1*
6) Vypočte $r = b^x \bmod n$
7) Vypočte $s = k.r + (x.h(M)) \bmod (n-1)$
8) Podpis zprávy M je dvojice (*r*, *s*)

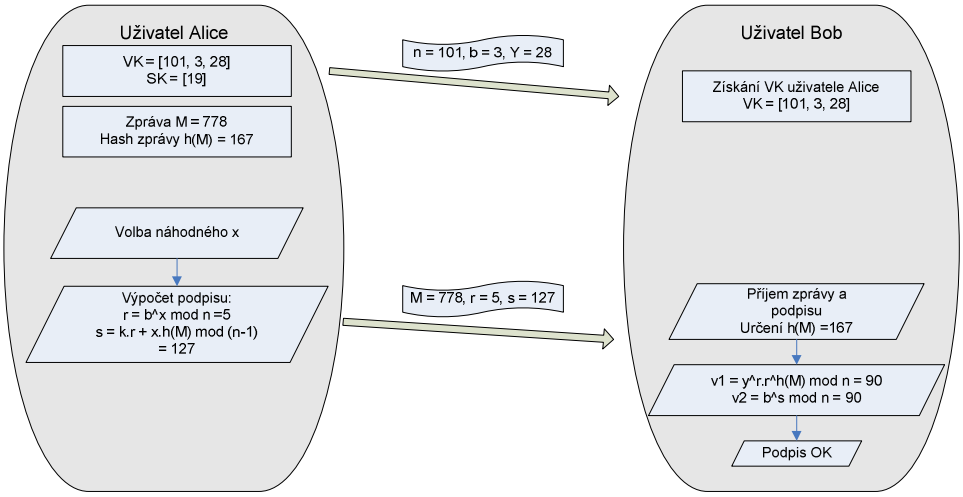
- parametr h(M), tzv. hash zprávy, je možné vypočítat řadou metod. Zde bude dále uvedena metoda vycházející ze Shamirova hashovacího algoritmu. Její popis, viz dále v textu.

**Ověření podpisu zprávy:**

6) Z důvěryhodného zdroje si obstará veřejný klíč uživatelky Alice (*n*, *b*, *y*)
7) Ověří, že $1 \le r \le n-1$, pokud toto není splněno, podpis automaticky označí za neplatný
8) Vypočte *h* (M) a $v_1 = y^r r^{h(m)} \bmod n$
9) Vypočte $v_2 = b^s \bmod n$
10) Za předpokladu, že $v_1 = v_2$, podpis přijme.

Stejně jako v případě ElGamalova kryptosystému pro šifrování zpráv, tak i v tomto případě je nutné pro každou podepisovanou zprávu volit odlišné a náhodné číslo *x*, aby se udržela základní bezpečnost podpisu.

Komunikace pak probíhá podle schématu naznačeného na obrázku. Pro bezpečnou distribuci klíče je třeba použít platného certifikátu, aby se zabránilo možnosti povržení zprávy.




## Shamirova hashovací funkce

**Teorie:**

Hashovací funkce je takzvaný jednosměrný otisk zprávy, tj. jedinečný kód, hash, vzniklý z původní zprávy použitím speciální, hashovací, funkce. Ze získaného hashe není obecně možné žádným způsobem obnovit zprávu, ze které byl vytvořen. Pokud se podaří najít jinou zprávu než původní, která má stejný hash, dochází ke kolizi. Pokud se takové zprávy daří nalézat cíleně, pak se hashovací funkce stává nepoužitelnou.

Shamirova hashovací funkce je jednoduchý návrh, jejhož odolnost vůči kolizím je dána obtížností řešení diskrétního logaritmu a obtížností faktorizace velkých čísel. Výhodou této funkce je zejména její jednoduchost a prokazatelná odolnost proti kolizím.

**Výpočet hashe:**

Nechť $n = p.q$ kde čísla $p$ a $q$ jsou velká prvočísla, takže faktorizace čísla $n$ je časově extrémně náročná.

Nechť $b$ je prvkem nejvyššího řádu v $\mathbf{Z}_n^*$ (tedy prvkem řádu $\lambda(n)=\text{lcm}(p-1, q-1)$). (pozn. Lcm – lowest common multiplier, nejmenší společný násobek. Například lcm(4, 6) = 12.)

Předpokládá se, že $n$ a $b$ jsou daná a veřejně známá čísla, $p$ a $q$ jsou tajná. Potom jestliže M je vstupní hodnota (zpráva), která má být hashována, vyjádřená jako nezáporné celé číslo libovolné hodnoty (i výrazně větší než je $n$), je hashovací funkce definována jako

$$\text{hash}(x) = b^x \bmod n\,.$$

**Otázky a úkoly:**

1) Cílem podepisování zpráv není:

A) Zajistit důvěrnost zprávy při přenosu

B) Umožnit autorizaci zprávy

C) Umožnit ověření, zda byla zpráva pozměněna během přenosu

2) Hash zprávy se vytváří za účelem:

A) Je to nezbytná součást pro vypočtení podpisu zprávy, bez něho není nikdy podpis možné vypočítat

B) Po šifrovaném přenosu umožňuje ověřit, že dešifrovaná zpráva je autentická a nebyla během přenosu pozměněna. Vytváří krátký otisk zprávy pro potřeby podpisu

C) Jeho jediným účelem je zkrátit přenášenou zprávu

3) Určete poslední část veřejného klíče, když víte, že SK = {11} a VK = {37, 3, x}.

4) Víte-li, že Shamirův hashovací algoritmus využívá prvočísel $p = 3$ a $q = 5$, vypočtěte otisk zprávy M = 351

5) Víte-li, že $p = 3$ a $q = 5$ a zpráva je M=3, z toho co víte o cyklických grupách a opakování prvků v nich, najděte kolizní M´, které dá stejnou hodnotu hashe jako M. Zamyslete se, proč lze hash stejně používat, i když lze cíleně nalézt kolizi.

6) Určili jste klíč pro podepisování SK = {174} a VK = {257, 38, 178}. Určete podpis zprávy, jejíž hash je roven $h(M) = 56$. Náhodné číslo zvolte $x = 5$. (odpověď zapište jako r, s).

7) Přijali jste zprávu M = 52, Hashovací klíč {n1 = 77, b1=30}, podpis P = {r:103, s:18039} a VK = {257, 38, 178}. Ověřte, zda je autorem zprávy vlastník certifikátu, v němž jste obdrželi VK. (do formuláře vyplňte ano / ne ).

**Příklad výpočtu:**

Zadání: Podepište zprávu M = 208 ElGamalovým kryptosystémem. Řád multiplikativní grupy pro výpočet klíče zvolte $n$ = 257, pro výpočet hashe použijte HK = {*n1*:143, *b1*:60}. Ostatní parametry volte libovolně. Zkuste podpis i ověřit.

*Tvorba klíče*:

- volba $b$ = 38, $k$ = 174. Pak:

$$b^k \bmod n = 38^{174} \bmod 257 = 178$$

- klíče: SK = {174}, VK = {257, 38, 178}

*Výpočet hashe:*

- zpráva M = 208, HK = {*n1*:143, *b1*:60}

$$h(M) = b_1{}^M \bmod n_1 = 60^{208} \bmod 143 = 92$$

*Tvorba podpisu:*

- SK = {*k*:174}, hash zprávy *h(M)*=92, z VK: $n$ = 257, $b$ = 38
- zvolen náhodný parametr $x$ = 13

$$r = b^x \bmod n = 38^{13} \bmod 257 = 103$$

$$s = k.r + (x.h(M)) \bmod (n-1) = 174.103 + (13.92) \bmod 257 = 18094$$

- podpis P = {r:103, s:18094}

*Ověření podpisu:*

- přijatá zpráva: M = 208, HK = {*n1*:143, *b1*:60}, VK = {257, 38, 178}, P = {r:103, s:18094}
  - podmínka: $r < n - 1$ ….. OK

$$h(M) = b_1{}^M \bmod n_1 = 60^{208} \bmod 143 = 92$$

$$v_1 = y^r r^{h(m)} \bmod n = 178^{103}.103^{92} \bmod 257 = 178$$

$$v_2 = b^s \bmod n = 38^{18094} \bmod 257 = 178$$

$v_1 = v_2$ …. Ok, autorem zprávy je vlastník přijatého certifikátu a zpráva je ok.

# Příloha 5: Texty k modulu Diffie-Hellmanův protokol

**Teorie:**

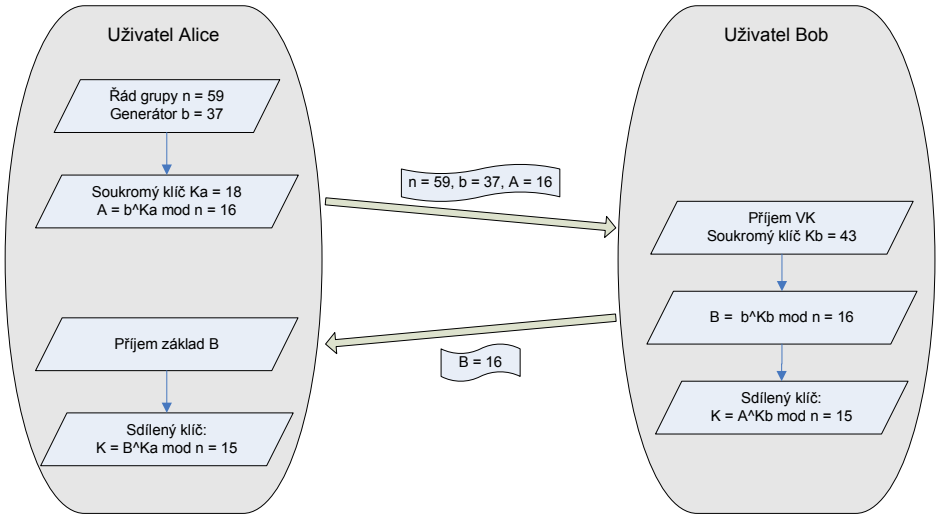
Diffie-Hellmanův protokol je protokolem určeným pro ustavení klíče mezi komunikujícími uživateli po veřejných nezabezpečených sítích bez toho, aby uživatelé mezi sebou před zahájením komunikace sdíleli jakékoli tajemství. Existuje v mnoha mírných modifikacích, jeho bezpečnost je však vždy založena na problému diskrétního logaritmu.

**Postup komunikace:**

6) Uživatel A zvolí velké prvočíslo *n* jako řád multiplikativní grupy $\mathbf{Z}_n^*$ a generátor *b* z této grupy, platí $1 \le b \le n-2$. Hodnota *n* musí být taková, aby hodnota *n-1* měla alespoň jeden velký prvočíselný dělitel nebo *n-1= 2p*, kde *p* je prvočíslo. Hodnoty *n* a *b* jsou veřejné.
7) Uživatel A z vybrané grupy zvolí svůj soukromý klíč *x* pro který platí ($1 \le x \le n-2$) a vypočte parametr A jako $A = b^x \bmod n$.
   Vypočtenou hodnotu A odešle nezabezpečeným způsobem uživateli B spolu s hodnotou generátoru *b* a řádu grupy *n*.
8) Uživatel B si z dané grupy zvolí vlastní soukromý klíč *y* ($1 \le y \le n-2$) a vypočte parametr B jako $B = b^y \bmod n$. Vypočtenou hodnotu odešle nezabezpečeným kanálem uživateli A.
9) Uživatel A od B přijme zprávu *B (=$b^y$)* a vypočte sdílený klíč $K = \left(b^y\right)^x \bmod n$
10) Uživatel B přijal od uživatele A zprávu A *($b^x$)* a vypočte sdílený klíč $K = \left(b^x\right)^y \bmod n$.

Oba uživatelé si tedy dokážou vypočítat stejný klíč, jehož hodnotu nedokáže určit nikdo jiný, k tomu aby to dokázal, musel by z přenášené komunikace určit hodnoty *x* a *y*, tedy vyřešit diskrétní logaritmus. Průběh komunikace znázorňuje i přiložený obrázek.

Velkou slabinu algoritmu však představuje zranitelnost vůči útoku mužem uprostřed (Man in the middle). Pokud totiž takový útočník zachytí zprávu A, kterou uživatel A zasílá uživateli B spolu s řádem grupy *n* a generátorem *b*, pak si může zvolit vlastní soukromý klíč AX a uživateli poslat vlastní odpověď B', kterou si vypočte vlastním klíčem AX. S uživatelem B pak naváže komunikaci stejně jako to měl původně v plánu uživatel A. Tímto způsobem pak má klíč pro komunikaci s A a jiný klíč pro komunikaci s B. Zprávy, které si uživatelé A a B vyměňují tak vždy dešifruje, přečte si je a znovu je zašifruje příslušným klíčem.




**Otázky a úkoly:**

1) Vyberte tvrzení o Diffie-Hellmanově protokolu, které platí beze zbytku

   A) Protokol nevyžaduje žádné sdílené tajemství mezi uživateli před začátkem komunikace, je odolný proti útoku „man in the middle“ a slouží pouze k ustavení klíče

   B) Protokol vyžaduje předem sdílené tajemství mezi uživateli, není odolný proti útoku „man in the middle“ a vedle ustavení klíče definuje i postup šifrování

   C) Protokol nevyžaduje žádné sdílené tajemství mezi uživateli před začátkem komunikace, je zranitelný útokem „man in the middle“ a slouží pouze k ustavení klíče

2) Veřejný klíč Diffie-Hellmanova protokolu tvoří:

   A) Řád grupy *n*, hodnota generátoru *b* a vypočtené *A*

   B) Hodnota vypočteného parametru *A*

   C) Tento protokol nedefinuje veřejný klíč

3) Uživatel odeslal druhé straně hodnotu $A = 8$, $n = 13$, $b = 7$ a obdržel odpověď $B = 1$. Je možné, aby při dodržení všech podmínek druhý uživatel vypočetl B = 1?

   A) Ano

   B) Ne

4) Jaká hodnota soukromého klíče dává hodnotu B = 1? (v dané grupě)

5) Projděte si postup ustavení klíče v aplikaci a pokuste se na papír ustavit klíč mezi dvěma uživateli. Jako základ cyklické grupy zvolte číslo $n = 23$ a generátor $b = 8$. Předpokládejte, že první uživatel zvolil soukromý klíč x = 5 a druhý uživatel y = 14. Jakou hodnotu má sdílený klíč?

6) Posloucháte přenos informací na sdíleném médiu. Podařilo se vám zachytit část komunikace, která obsahuje informace o zvolených parametrech pro ustavení klíče Diffie-Hellmanovým protokolem. Víte, že n = 7, b = 3, parametr A prvního uživatele je A = 4 a parametr B druhého uživatele je B = 6. Pokuste se hrubou silou zjistit, jaká je hodnota sdíleného klíče?

**Příklad výpočtu:**
Dva uživatelé mezi sebou chtějí komunikovat pomocí symetrické šifry. Nemají však společný tajný klíč, proto se rozhodnou jej ustavit D-H protokolem. Jako základ cyklické grupy zvolí číslo $n = 257$ a generátor $b = 134$. Jak postupují dále?

a) *uživatel 1*:

- zvolí soukromý klíč $x = 37$
- vypočte parametr A: $A = b^x \bmod n = 134^{37} \bmod 257 = 67$
- uživateli 2 odešle $n = 257$, $b = 134$, $A = 67$

b) *uživatel 2:*

- obdrží $A = b^x \bmod n = 134^{37} \bmod 257 = 67$
- zvolí soukromý klíč $y = 211$
- vypočte $B = b^y \bmod n = 134^{211} \bmod 257 = 165$
- odešle $B = 165$
- vypočte sdílený klíč $K = A^y \bmod n = 67^{211} \bmod 257 = 117$

c) *uživatel 1:*

- obdrží $B = 165$
- vypočte $K = B^x \bmod n = 165^{37} \bmod 257 = 117$

# Příloha 6: Texty k modulu DSA a podepisování zpráv

**Teorie:**

Standard DSA je standard určený pro generování a ověřování digitálních podpisů zpráv. Někdy se zaměňuje s označením DSS (Digital Signature Standard), což je americký standard digitálního podpisu, který z DSA vznikl v roce 1991. DSS je s DSA v podstatě totožný, proto záměna označení není chybou. Rozdílem je, že DSS specifikuje maximálně 1024bitů dlouhý klíč, DSA takové omezení nemá.

Podpisové schéma opět využívá diskrétního logaritmu v konečné cyklické grupě $\mathbf{Z}_n^*$. Doposud nebylo prokázáno, že je to dostatečnou dostatečně bezpečný, stejně jako není prokázána bezpečnost RSA digitálního podpisu, není ale nalezena ani metoda, která by jej dokázala prolomit.

**Tvorba klíče:**

7) Zvolí prvočíslo *n* takové, že $2^{511+64t} < n < 2^{512+64t}$ ($0 < t < 24$) a prvočíslo *q*, které je celočíselným dělitelem *n-1* (zpravidla délky 160-512 bitů).
8) Zvolí parametr h takový, že *h* < *n-1* a současně $h^{(n-1)/q} \bmod n > 1$
9) Vypočte generátor *b* jako $b = h^{(n-1)/q} \bmod n$
10) Zvolí náhodné celé číslo k, pro které platí $1 < k < q$
11) Vypočte $y = b^k \bmod n$
12) Veřejný klíč je tvořen kombinací (*n*, *q*, *b*, *y*), privátní klíč je *k*.

**Tvorba podpisu:**

5) Zvolí se náhodné celé číslo x, pro které platí $0 < x < n$
6) Vypočte $r = (b^x \bmod n) \bmod q$
7) Vypočte $s = x^{-1}\{h(\mathrm{M}) + kr\} \bmod q$
8) Digitální podpis zprávy M je dvojice (*r*, *s*)

Pro vytvoření otisku (hashe) zprávy *h*(M) využívá DSA algoritmu SHA-1, který vytvoří 160 bitů dlouhou sekvenci dat, kterou je třeba pro potřeby výpočtu převést na číslo.

**Ověření podpisu:**

7) Uživatel získá z důvěryhodného zdroje veřejný klíč (*n*, *q*, *b*, *y*)
8) Ověří, že $0 < r < q$ a $0 < s < q$. Pokud není splněna jedna z podmínek, podpis odmítne
9) Vypočte $w = s^{-1} \bmod q$ a *h*(M)
10) Vypočte $u_1 = w.h(\mathrm{M}) \bmod q$ a $u_2 = r.w \bmod q$

11) Vypočte $v = (b^{u_1} y^{u_2} \bmod n) \bmod q$

12) Podpis přijme v případě, že $v = r$, v opačném případech podpis zprávy odmítne.

V současnosti se systém DSA stále hojně používá, postupně se však přechází k jeho vylepšené variantě ECDSA, která místo multiplikativních cyklických grup využívá aditivní grupy eliptických křivek. Pro bezpečnou distribuci veřejných klíčů je nutné využívat služeb některé z certifikačních autorit.

**Otázky a úkoly:**

1) Pokud víme, že běžná délka klíče je 1024 bitů, jak velká by měla být délka klíče certifikační autority?
   A) stačí 512 bitů
   B) také 1024 bitů
   C) 2048 a více
2) Lze při dnešních výpočetních výkonech považovat DSA a klíčem 1024 bitů za bezpečné?
   A) Ano, ale delší klíč by byl vhodný
   B) Ne, klíč je příliš krátký a DSA již není považováno za bezpečné
3) Lze vytvořit vlastní certifikát pro hodnověrnou distribuci klíčů bez certifikační autority?
   A) Ano, běžně se to tak dělá a je to v pořádku
   B) Ne, certifikační autorita musí podepsat certifikát svým klíčem
4) Co je to certifikát?
   A) Certifikační autoritou zašifrovaný veřejný klíč
   B) Soubor s osobními údaji uživatele s certifikační autoritou podepsaným hashem těchto údajů
   C) Potvrzení, že uživatel smí používat služeb zabezpečeného přenosu

**Příklad výpočtu:**

Podepište zprávu, jejíž hash *h*(M)=39. Parametry zvolte dle potřeby.

*Tvorba klíče:*

- zvolí parametr *n* = 53 a *q* = 13. Ověří podmínku: (*n*-1)/*q* = celé číslo 52/13=4 … ok

- zvolí parametr *h* = 7, ověří že $h^{(n-1)/q} \bmod n > 1 \ldots 7^4 > 1$ … ok

- vypočte generátor $b = h^{(n-1)/q} \bmod n = 7^{52/13} \bmod 53 = 16$

- zvolí si *k* = 8, ověří že $1 < k < q$ … 1<8<13 … ok

- vypočte $y = b^k \bmod n = 16^8 \bmod 53 = 42$

- dvojice klíčů je: SK = {*k*:8}, VK = {*n*:53, *q*:13, *b*:16, *y*:42}

*Tvorba podpisu:*

-zvolí x = 21, ověří podmínku $0 < x < n$ … 0 < 21 < 53 … ok

-vypočte: $r = (b^x \bmod n) \bmod q = (16^{21} \bmod 53) \bmod 13 = 3$

$$s = x^{-1}\{h(\mathrm{M}) + kr\} \bmod q = 3^{-1}(39 + 8.3) \bmod 13 = 3$$

*Ověření podpisu:*

- přijme podpis P = {r:3, s:3}, hash by musel vypočítat, zde uvažuje, že ví, že *h*(M)=39

- vypočte: $w = s^{-1} \bmod q = 3^{-1} \bmod 13 = \frac{1}{3}$

$$u_1 = w.h(\mathrm{M}) \bmod q = \frac{39}{3} \bmod 13 = 0$$

$$u_2 = r.w \bmod q = \frac{3}{3} \bmod 13 = 1$$

- dále vypočte: $v = (b^{u_1} y^{u_2} \bmod n) \bmod q = (16^0.42^1 \bmod 53) \bmod 13 = 3$

- protože *r* = *v*, podpis přijme